Thesis for The Degree of Doctor of Philosophy

Living Boundary Objects to Support Agile Inter-Team Coordination at Scale

Rebekka Wohlrab

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2020

Living Boundary Objects to Support Agile Inter-Team Coordination at Scale

Rebekka Wohlrab

ISBN 978-91-7905-269-0
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 4736.
ISSN 0346-718X

Technical Report No 183D
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2020.

Abstract

Context: In the last decades, large-scale agile development has received increasing attention, as also organizations with many stakeholders and large systems aim for higher development speed and focus on customer value. A recognized research challenge in large-scale agile development relates to inter-team coordination. To coordinate effectively, organizations need to identify what knowledge is required across team borders and how it can be managed over time. Knowledge is potentially manifested in *boundary objects*—artifacts that create a shared understanding between teams (e.g., requirements or architecture descriptions). Traceability between artifacts is a key necessity to manage change in agile contexts. Moreover, agile practitioners aim to reduce the documentation effort to absolutely crucial artifacts and trace links.

Objective: This thesis aims to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. We focus especially on how knowledge can be made explicit in artifacts and trace links that are evolved over time.

Method: We empirically investigated problems and developed solutions using a research approach that was inspired by design science. Case studies, an in-depth design science study, a mixed methods study, and surveys were performed. Using this mix of research methods, we leveraged both qualitative and quantitative data.

Results: We coined the concept of *living boundary objects* to manage knowledge for inter-team coordination. Living boundary objects are boundary objects that are traced to other artifacts, kept up to date, and serve for inter-team coordination. They should be established early in the lifecycle to create a common understanding of the product to be developed. We scrutinized architecture descriptions, interfaces, and requirements and traceability information models as examples of concrete boundary objects. We recommend establishing alignment using a common high-level structure, but also supporting diverse knowledge management practices to fulfill the individual needs of agile teams.

Conclusions: Our contributions help to establish knowledge management practices that are considered beneficial by practitioners and focus on the crucial aspects to align agile teams on. We suggest concepts and requirements for knowledge management tools that take the distinct role of living boundary objects into consideration and can be adjusted as organizations' needs evolve.

Keywords

large-scale agile development, boundary objects, empirical software engineering, traceability management

Acknowledgments

I am extremely grateful to many people for their support during the last four years. I would like to thank my supervisor Patrizio Pelliccione and my co-supervisor Eric Knauss who both have been very encouraging and positive, provided me with valuable advice, and helped me sharpen my conclusions. I am also grateful to my examiner Jan Bosch who has given me the freedom to conduct independent research and has been very supportive.

I would also like to thank Darja Šmite, Casper Lassenius, Per Runeson, and Jelena Zdravkovic for accepting to serve on my committee and providing me with feedback on earlier versions of the thesis.

I want to thank my colleagues and the administrative staff at the Software Engineering Division for the supportive and inspiring work environment. Thanks to all my co-authors, in particular to Jan-Philipp Steghöfer, Jennifer Horkoff, Rashidah Kasauli, Salome Maro, and Tony Anjorin for the enjoyable and fruitful collaboration on traceability, requirements engineering, and boundary objects. I would like to thank Rogardt Heldal and Ulf Eliasson for our interesting discussions and collaboration on architecture-related topics. I am grateful to my fellow PhD students for their encouragement, friendship, and the exchange of experiences, especially to Katja Tuma, Magnus Ågren, Piergiuseppe Mallozzi, and Sergio García.

A big '*thank you*' to all individuals who participated in my studies—your involvement was essential for this PhD endeavor! In particular, I would like to thank Anders Alminger for sharing his insights on agile architecture and supporting me with Paper D, as well as Ali Shahrokni for many inspiring conversations and the enjoyable collaboration on Paper F. I am also grateful to all employees of Systemite AB for providing a splendid environment for me as an industrial PhD student. I thank Mats Larsson and Fredrik Berglund for their support, great discussions, and comments on drafts of this thesis.

I would like to express my gratitude to my friends and family, especially Jan, Marta, PG, and Bill. I would also like to thank Lovisa, Henrik, Eivor, Anders, Emilia, Karin, Maria, and Ylva for your prayers and encouragement. Special thanks go to my father Richard, my mother Sabine, and my brother Lukas, as well as Marcus for your love, support, and many happy moments.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

List of Publications

Appended publications

This thesis is based on the following publications:

[A] R. Wohlrab, P. Pelliccione, E. Knauss, M. Larsson.
"Boundary Objects and Their Use in Agile Systems Engineering"
Journal of Software: Evolution and Process; 31:e2166, 2019.

[B] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, P. Pelliccione.
"Collaborative Traceability Management: A Multiple Case Study from the Perspectives of Organization, Process, and Culture"
Requirements Engineering, 2018.

[C] R. Wohlrab, U. Eliasson, P. Pelliccione, R. Heldal.
"Improving the Consistency and Usefulness of Architecture Descriptions: Guidelines for Architects"
Proceedings of the International Conference on Software Architecture (ICSA'19), 2019. ***Best Paper Award.***

[D] R. Wohlrab, P. Pelliccione, E. Knauss, R. Heldal.
"On Interfaces to Support Agile Architecting in Automotive: An Exploratory Case Study"
Proceedings of the International Conference on Software Architecture (ICSA'19), 2019.

[E] R. Wohlrab, E. Knauss, P. Pelliccione.
"Why and How to Balance Alignment and Diversity of Requirements Engineering Practices in Automotive"
Journal of Systems and Software; 162:110516, 2020.

[F] R. Wohlrab, P. Pelliccione, A. Shahrokni, E. Knauss.
"Why and How Your Traceability Should Evolve: An Automotive Perspective"
Revised version submitted to IEEE Software, 2020.

Other publications

The following publications were published during my PhD studies. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, A. Anjorin.
"Collaborative Traceability Management: Challenges and Opportunities"
Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), 2016.

[b] S. Maro, A. Anjorin, R. Wohlrab, J.-P. Steghöfer.
"Traceability Maintenance: Factors and Guidelines"
Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16), 2016.

[c] R. Wohlrab.
"Continuous Management of Design- and Run-Time Artifacts for Self-Adaptive Systems"
Proceedings of the 39th IEEE/ACM International Conference on Software Engineering Companion (Doctoral Symposium, ICSE'17), 2017.

[d] R. Wohlrab, P. Pelliccione, E. Knauss, S. C. Gregory.
"The Problem of Consolidating RE Practices at Scale: An Ethnographic Study"
Proceedings of the 24th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'18), 2018.

[e] R. Wohlrab, P. Pelliccione, E. Knauss, M. Larsson.
"Boundary Objects in Agile Practices: Continuous Engineering of Systems Engineering Artifacts in the Automotive Domain"
Proceedings of the International Conference on Software and System Processes (ICSSP'18), 2018. ***Best Paper Award (Research Track)****.*

[f] E. Knauss, G. Liebel, J. Horkoff, R. Wohlrab, R. Kasauli, F. Lange, P. Gildert.
"T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development"
Proceedings of the 26th IEEE International Requirements Engineering Conference (RE'18), 2018.

[g] S. García, P. Pelliccione, C. Menghi, T. Berger, R. Wohlrab.
"An Architecture for Decentralized, Collaborative, and Autonomous Robots"
Proceedings of the IEEE International Conference on Software Architecture (ICSA'18), 2018.

[h] J.-P. Steghöfer, E. Knauss, J. Horkoff, R. Wohlrab.
"Challenges of Scaled Agile for Safety-Critical Systems"
Proceedings of the 20th International Conference on Product-Focused Software Process Improvement (PROFES'19), 2019. ***Best Paper Award****.*

[i] R. Kasauli, R. Wohlrab, E. Knauss, J.-P. Steghöfer, J. Horkoff, S. Maro.
"Charting Coordination Needs in Large-Scale Agile Organizations with Boundary Objects and Methodological Islands"
Proceedings of the International Conference on Software and System Processes (ICSSP'20), 2020.

[j] R. Wohlrab, A. Anjorin, A. Shankar Mishra.
"What do Users Expect of Bidirectional Transformations?"
Proceedings of the European Conference on Modelling Foundations and Applications (ECMFA'20), 2020.

Research Contribution

For all appended publications, I assumed the main responsibility for the design of the research methods, data collection, data analysis, discussion, and writing. For Paper C, the design of a first survey was performed by my co-authors, as well as the data collection and initial data analysis. These steps were extended by me with further analysis, the planning, execution, and analysis of a second survey, as well as writing and editing the paper.

Contents

Chapter 1

Introduction

Agile development methods have been increasingly adopted over the last years [1, 2], as companies aim for higher flexibility to adapt to changing requirements and market needs. In the annual "*State of Agile*" survey of 2019, 97% of the participants indicated that they use agile development methods [3]. Although the "*State of Agile*" survey is not a scientific survey, it indicates that agile methods are in fact widely spread. Agile methods were initially adopted in small and co-located teams [4], but have become more used in larger companies as well [5, 6].

Large-scale agile development typically involves a large number of actors, as well as multiple systems with interdependencies between them [7]. Many approaches for large-scale agile development are based on assumptions that hold in small-scale agile scenarios, but not in large development contexts. In particular, there is a problematic assumption that agile practices can be linearly scaled up [7]. For instance, the use of Scrum-of-Scrum meetings to discuss decisions and propagate them back to the teams comes with problems (e.g., low perceived usefulness and efficiency) [8]. Mechanisms are needed to establish knowledge sharing and inter-team coordination as an ongoing activity between actors [7], especially over a large geographical and temporal distance [9].

To address this issue, the challenge of inter-team coordination has been included in the research agenda of large-scale agile software development [5, 10]. To enable coordination in large-scale contexts, knowledge related to systems, interdependencies, and activities of actors needs to be shared [11, 12]. Sharing tacit knowledge through face-to-face conversations plays a crucial role in knowledge management in agile contexts, but also externalizing knowledge and capturing it in knowledge artifacts should not be neglected [13].

Externalized knowledge is especially important where employee turnover rates are high and new sites are established [12, 14]. A core component of (process) knowledge management is traceability, i.e., the ability to create and use links between knowledge artifacts [15]. The development of a traceable web of linked artifacts and people can support organizations in creating, storing, retrieving, transferring, and applying knowledge. However, practitioners in globalized agile software development often struggle with identifying what knowledge is needed and should be made explicit in artifacts and trace links [16–18]. More empirical studies are needed to support inter-team coor-

dination and manage knowledge in suitable ways [5, 8].

This thesis' goal is to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. We focus mainly on the use of externalized knowledge (e.g., artifacts or trace links) and surrounding mechanisms for the technical coordination of development activities [11]. Our subgoals are

(G1) to provide empirical insights into the current state of knowledge management for inter-team coordination in large-scale agile development and
(G2) to explore solutions to support knowledge management for inter-team coordination in large-scale agile development.

Concretely, to address G1, we create an empirical basis to conceive approaches for knowledge management in large-scale agile development, by focusing on current practices and challenges. Addressing G2, we created solutions in the form of guidelines and principles to support knowledge management, as well as concepts for tool solutions.

In particular, we coin the concept of *living boundary objects* that can be leveraged to manage knowledge for inter-team coordination. Boundary objects are artifacts used by groups of actors to create a common understanding while being adaptable to the specific needs of each group [19]. In this thesis, we define living boundary objects as boundary objects that are traced to other artifacts, kept up to date, and serve for inter-team coordination.

This cumulative thesis is built on an introductory chapter, followed by Chapters A–F containing the respective appended publications Papers A–F. The remainder of this introductory chapter is structured as follows: Section 1.1 introduces the background of knowledge management and inter-team coordination in large-scale agile development. The goals of this thesis are described in further detail in Section 1.2 and positioned in the context of related work in Section 1.3. The research approach is described in Section 1.4, followed by a summary of the main contributions of this thesis in Section 1.5. Section 1.6 gives high-level answers to this thesis' research questions. We discuss our findings in Section 1.7 and conclude this thesis in Section 1.8.

1.1 Background

This section introduces central concepts relevant to this thesis. We describe large-scale agile development in Section 1.1.1, inter-team coordination in Section 1.1.2, knowledge management in Section 1.1.3, traceability in Section 1.1.4, and boundary objects in Section 1.1.5.

1.1.1 Large-Scale Agile Development

Agile software development methods have been used since the early 2000s [20]. The agile manifesto [21] coined important principles: people and interactions between them were described as the cornerstones of agile development, as well as working software, customer collaboration, and fast response to change. Typically, agile methods involve cross-functional teams of individuals with different expertise that work in short iterations (e.g., sprints) and rely on

face-to-face communication. A plethora of agile methods has been developed over time [20], e.g., Scrum or eXtreme Programming. For the development of physical products, agile methods are perceived to improve communication, reaction time to changes, flexibility, transparency, and commitment [22].

It was acknowledged early on that ways are needed to scale agile methods to larger development contexts (e.g., [20, 23]), which led to the research area of *large-scale agile development* [24]. An agile development context is called *large-scale* when between two and nine teams are involved [25] and we use that definition in this thesis. Other definitions refer to the scope of the developed system(s), team size, and project duration [7, 26]. In fact, most companies we collaborated with had more than 9 teams, which makes them even *very large-scale* [25].

To scale agile methods, a number of frameworks have been developed, for instance, the Scaled Agile Framework (SAFe) [27] or Large-Scale Scrum (LeSS) [28]. There exist several reported benefits of using agile scaling frameworks for large-scale agile development [29], e.g., increased efficiency, motivation, and quality. Some frameworks are rather lightweight and built on Scrum, e.g., Scrums of Scrums or LeSS. SAFe has been listed as the most complex framework [29], as it is rather process-heavy, introduces stronger changes to the organizational structure, and relies on a larger number of artifacts. Agile scaling frameworks commonly scale existing practices up in a linear way. In LeSS, for instance, sprint planning or product backlog refinement activities are used on a high level for all teams, as well as individually for each team. SAFe establishes so-called *agile release trains* to structure teams of agile teams, as well as backlogs on several levels (e.g., *program, solution, and team backlogs*). Product-related knowledge is intended to be distributed by key roles. For instance, enterprise, solution, and system architects are roles that manage architectural knowledge in SAFe [27].

When adopting agile methods in large-scale development, inter-team coordination and knowledge sharing require specialized approaches [8, 30]. We describe the theoretical background related to inter-team coordination and knowledge management in the following sections.

1.1.2 Inter-Team Coordination

When engineering complex products of realistic size, humans need to work together. It is not feasible to design, develop, test, operate, and maintain such products without involving several experts that collaborate. Collaboration is defined as the "*process in which two or more agents work together to achieve shared goals*" [31]. Coordination, the "*management of interdependencies between activities*" [32], is necessary to achieve the goal of collaboration. As activities are explicitly or implicitly connected to real-world objects, the management of dependencies between such objects is another aspect of coordination, e.g., between components of a system [32].

Coordination mechanisms are defined as "*the organizational arrangements that allow individuals to realize a collective performance*" [33]. When humans aim to collaborate in software or systems engineering, they typically divide the product into smaller parts that can be developed by groups or teams. These groups are typically characterized by different backgrounds and sub-disciplines

(e.g., software engineering, mechanical engineering, or electrical engineering). Agile companies aim for increased development speed and flexibility to change. To this end, autonomous agile teams should be empowered to provide customer value and tailor development practices to their needs. At the same time, teams are required to integrate their output into one common, deliverable product and align their work with a large number of heterogeneous stakeholders [34]. In large-scale agile development, these stakeholders typically belong to different organizations with different goals. The required balance between empowerment and alignment and the variety of heterogeneous stakeholders make agile inter-team coordination a challenging activity.

Several approaches to support coordination exist. They can be classified as impersonal, personal, and group mechanisms [35]. In the impersonal mode, knowledge is typically codified in artifacts and used for coordination, whereas personal and group mechanisms involve direct interactions between humans, either as individuals or as a team. In large-scale agile frameworks, coordination is supported by several mechanisms: for instance, by backlogs that help to coordinate activities for the coming iteration, face-to-face communication and planning meetings, or specialized roles (e.g., architects) that coordinate concerns across team borders (see Section 1.1.1).

1.1.3 Knowledge Management

Shared knowledge is needed to support coordination, both in the form of dynamic, fleeting knowledge, as well as long-lasting knowledge [11, 36]. Knowledge is understood as the most abstract layer in the hierarchy of data, information, and knowledge [37]. In the field of knowledge management, researchers distinguish between tacit, implicit, and explicit knowledge [38]. Tacit knowledge defies codification and cannot easily be expressed. Implicit knowledge can be codified in knowledge artifacts, however, its meaning is not explicitly captured, but can be inferred by the reader. Explicit knowledge is captured in knowledge artifacts with clearly defined semantics and its meaning can be transferred between people.

Knowledge management can be used to coordinate an organization with the goal of providing value through reuse and innovation:

> *Knowledge management is the deliberate and systematic coordination of an organization's people, technology, processes, and organizational structure in order to add value through reuse and innovation. This coordination is achieved through creating, sharing, and applying knowledge as well as through feeding the valuable lessons learned and best practices into corporate memory in order to foster continued organizational learning.* (Kimiz Dalkir [39])

In agile environments, tacit knowledge plays an essential role [40]. Typically, face-to-face meetings are used to exchange tacit knowledge. In large-scale agile development, it is not feasible to only rely on this form of knowledge management. To this end, artifacts are instrumental in managing knowledge in software engineering and agile software development [41, 42]. An artifact is "*a self-contained work result, having a context-specific purpose and constituting a physical representation, a syntactic structure and a semantic content*" [42].

For instance, a handwritten sketch of an architecture model, a text document containing requirements, or a file containing C code can be artifacts. Artifacts have specific types and typically, versions of artifacts are created and controlled in specialized tools. Artifacts and documentation are needed for record-keeping and coordination between separate development teams in large-scale agile development [43] and traceability between these artifacts needs to be established and managed [42].

Knowledge management cycles have been suggested and extended over the last decades [39]. They use different terms and have different focus areas, but typically include the main areas of capturing knowledge, sharing it in an organization, using it, and assessing whether new knowledge should be captured. In the following, we introduce Dalkir's integrated knowledge management cycle [39], depicted in Figure 1.1. The *capture and/or creation* phase involves identifying new or existing knowledge that will be codified in artifacts and trace links between them. After having created or captured knowledge in artifacts and trace links, their content is assessed to identify what knowledge and know-how are important to share. In the next step, knowledge is *shared and disseminated* to relevant users. It involves making knowledge available and searchable, establishing communities of practice that can share tacit knowledge [39, 40], and creating official or unofficial knowledge networks. In many organizations, the assumption exists that available knowledge will be used. To be *acquired and applied*, knowledge needs to be contextualized. Contextualization involves tailoring knowledge to particular disciplines and persons and their information needs, so that people can leverage it. Finally, the artifacts and trace links are updated based on gathered experiences from knowledge acquisition and application and new knowledge is captured.

Knowledge management can be left up to individuals or be prescribed by organizations to varying degrees. Balancing the fluid, dynamic and the institutional, controlled domains of knowledge management is a core challenge [44]. To maintain the right balance, just-enough-discipline (JED) is needed [45]. We focus on the level of discipline, alignment, or rigor in Papers A and E.

Knowledge management and coordination can be studied from different angles and are motivated by a variety of drivers [11, 46], e.g., task-related, people-related, or organizational drivers. In the context of this thesis, we are mainly interested in technology-related aspects (e.g., how to use collaborative technologies and artifacts for knowledge sharing and coordination) and their use for technical coordination of development activities [11, 46]. To this end, we focus on traceability and boundary objects as facilitators of knowledge management.

1.1.4 Traceability

Traceability is a powerful enabler of knowledge management [15, 47], as trace links capture knowledge about relations between artifacts and can support knowledge management activities (e.g., quickly identifying relevant knowledge artifacts impacted by a proposed change). Software or systems traceability is the ability to create and use links between artifacts [48]. These links are called *trace links* and create a connection from a *source artifact* to a *target artifact*. (Theoretically, also n:m trace links can exist, but we refer to 1:1 relations in this

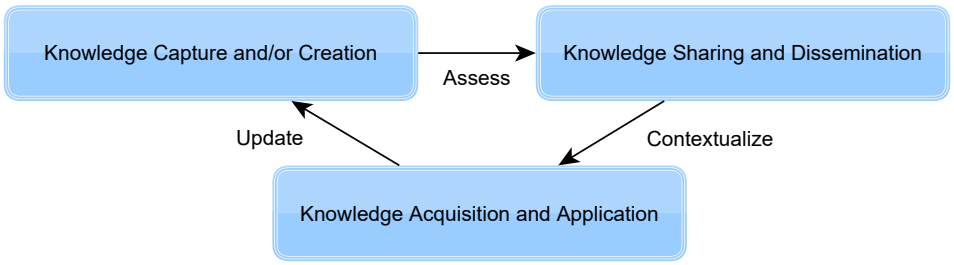

Figure 1.1: An integrated knowledge management cycle [39]

thesis.) They can have types (e.g., "*implements*" or "*tests*"), can be versioned, and used for several purposes. Change impact analysis, demonstration of compliance with standards, coverage analysis, and dependency analysis are common tasks that are supported by traceability [49]. Trace links need to be created, maintained, and used by collaborating stakeholders.

Planning and managing a traceability strategy is necessary to ensure that traceability is managed in a beneficial way [48]. Figure 1.2 shows the associated activities: based on an identification of stakeholders' needs, it is analyzed what resources are needed and what cost and benefit of traceability can be expected. Once the resources are allocated, a traceability information model is defined, as well as processes and tooling. A traceability information model (TIM) is a graph defining permissible artifact types, trace link types, and their relationships. A TIM can also contain information regarding the cardinality of the artifacts connected through a trace link, the direction of trace links, trace link semantics, main creators, etc. After having planned the traceability strategy, stakeholders implement the plan by creating, maintaining, and using trace links in actual instances. Based on the instantiated data, it is assessed whether the traceability strategy addresses stakeholders' needs. Typically, the described activities are executed in an ongoing cycle.

It should be noted that the described activities can be related to the integrated knowledge management cycle described before (see Figure 1.1): knowledge regarding traceability needs and resources is captured or created, then an information model, processes, and tooling are planned for, created, shared, and disseminated to inform stakeholders about the strategy, and finally, it is acquired and applied in the implementation step and assessed to understand the need for refinement.

More often than not, activities related to traceability require multiple stake-

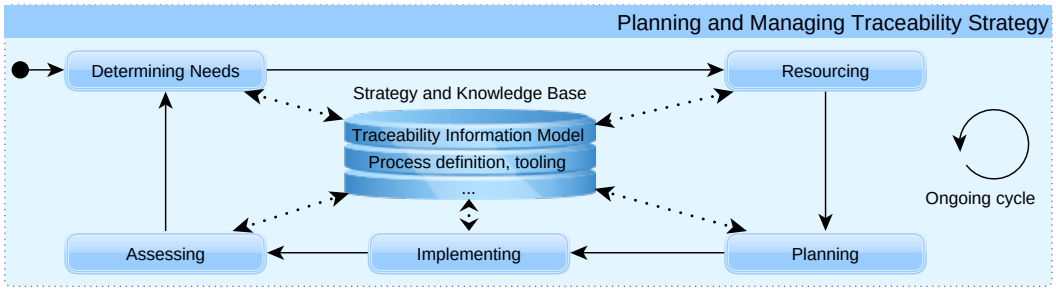

Figure 1.2: Planning and managing a traceability strategy, adapted from [48]

holders. To this end, we define collaborative traceability management in Paper A as "*the collaborative planning, organization, and coordination of all tasks concerned with traceability in multi-person projects across organizational, discipline, or tool boundaries.*"

1.1.5 Boundary Objects

We make use of theoretical concepts from *Actor-Network Theory* [50,51] in this thesis, as it helps to understand large-scale settings as an interdisciplinary network of various actors and roles. Using actor-network theory is beneficial, as "*co-ordination is a strategic term that hints at the existence of a centred strategist, someone with an overview*" [52]—and this assumption does not always hold in large companies. Instead, attempts to align and coordinate work are initiated simultaneously by several actors.

Actor-network theory was first coined by Callon [50] and Latour [51]. It is concerned with heterogeneous networks of human and non-human actors, organizations, and standards. Actors have individual interests and use so-called *translations* to consolidate their concerns. The end result of a successful translation is that one actor serves as a "*spokesman*" for the concerns of a group of actors [50]. Typically, multiple actors initiate translations at the same time [19].

In a network of actors, certain artifacts emerge in situations where multiple translations happen in parallel [19]. These artifacts are *boundary objects* and they are used by different groups as a means of translation, but also enable actors' autonomy. The definition by Star and Griesemer is as follows:

> *Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites.* (Susan Leigh Star and James R. Griesemer [19])

The parties employing boundary objects are actors or organizational groups, potentially with different disciplines and backgrounds. They are located at different sites, not merely in the geographical sense, but also considering temporal, cognitive, or psychological distance [53]. Being "*plastic enough to adapt to local needs*" relates to the interpretive flexibility of boundary objects, which allows different actors to freely interpret boundary objects and leverage them for their purposes [54]. A "*common identity across sites*" implies a shared understanding of common aspects to agree on, e.g., concerning a system, organization, technology, processes, or other areas in which coordination is needed. Boundary objects are a means of facilitating coordination with impersonal communication [55].

In Paper A, we present additional aspects related to boundary objects in large-scale agile development: *vertical boundary objects* are boundary objects that are used across engineering areas (e.g., function specification, analysis, and design), whereas *horizontal boundary objects* are used between teams working on the same level of abstraction and typically in the same phase (e.g., a signal used between two components developed by different teams). Humans can feel tempted to devise boundary objects upfront [56] and create so-called *designated boundary objects*, rather than *boundary objects-in-use* [57]. Ideally,

boundary objects are traced to and from other artifacts, so that changes in a boundary object can be propagated to others and a common understanding or identity across team borders can be kept. In this thesis, we use the term "*living boundary objects*" for boundary objects that are traced to other artifacts, kept up to date, and serve for inter-team coordination.

1.1.6 SystemWeaver

This PhD project has been conducted in close collaboration with Systemite AB, a company developing the systems engineering tool SystemWeaver[1]. It is mainly used in automotive companies to support requirements engineering, design, safety analysis, testing, and maintenance activities. SystemWeaver can be customized to users' needs, supported by an adjustable underlying metamodel/information model, and customizable visualizations of data. Data can be presented in reports, graphs, diagrams, tables, or export formats (e.g., ReqIF[2]). Moreover, SystemWeaver is a collaborative modeling platform that allows teams to work together online and in real-time. Versions of artifacts and trace links can be created in SystemWeaver. An artifact is either "*in work*", i.e., not yet stable, or at a specific version. Information is stored for each creation and change, recording the user, date, and time. For each artifact, a responsible owner is defined. In the metamodel, artifact types and trace link types are specified. All trace links have a type and general relations can be created using the type "*reference*." SystemWeaver is used by thousands of users at multiple companies. Several of these companies participated in our studies and allowed us to collect empirical data. Moreover, this thesis contributes to the conception of future tool solutions related to SystemWeaver, as we present in Section 1.6.

1.2 Goals of the Thesis

In this section, we revisit the goals of this thesis. As stated in the introduction, the main goal is to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. We address this goal by providing empirical insights into the topic and exploring solutions to improve how knowledge can be managed. The focus lies on *explicit knowledge* for inter-team coordination in large-scale agile development. Note that the scope is limited to software and systems engineering teams and to the coordination of development activities.

Our subgoal (G1) is *to provide empirical insights into the current state of knowledge management for inter-team coordination in large-scale agile development.* To this end, we answer the following research question:
RQ1: What is the current state of managing knowledge for inter-team coordination in large-scale agile development?

Our subgoal (G2) is *to explore solutions to support knowledge management for inter-team coordination in large-scale agile development.* We answer the following research questions:

[1] `https://systemweaver.se/`
[2] `https://www.omg.org/reqif/`

RQ2: What guidelines can help practitioners to manage knowledge for inter-team coordination in large-scale agile development?
RQ3: How can tool solutions support practitioners to manage knowledge for inter-team coordination in large-scale agile development?

While implicit or tacit knowledge plays a role in many of our studies, we focus primarily on how knowledge is captured, shared, and traced in explicit ways (using artifacts and trace links). In particular, we focus on artifacts serving as boundary objects in large-scale agile organizations. With respect to the current state (**RQ1**), we examine lifecycle phases of artifacts and trace links, from their initial creation to changes, phases of stability, and deprecation. We focus on how knowledge is captured and applied, what supporting coordination mechanisms are used, how companies aim to align teams while supporting diverse practices, and how individuals can be motivated to invest in knowledge management. We investigate solution candidates in the form of guidelines (ranging from rather abstract principles to concrete suggestions for practices) and tool solutions (**RQ2** and **RQ3**). Figure 1.3 shows how our six papers provide answers to our main research questions. All papers contribute to the analysis of the current state (RQ1). Papers A and B lay the foundation for further studies in Papers C–F. Papers A–E present guidelines (RQ2) and Papers A and F describe tool solutions (RQ3). In Section 1.4, we describe the research approach and focus of our papers in further detail.

1.3 Related Work

As this PhD thesis covers a considerably broad topic, there exists a large body of knowledge related to each of our research questions. We refer to the related work sections of Papers A–F for details and provide an overview of related work in this section.

The importance of using knowledge management for inter-team coordination has been stressed by previous research, since successful self-organizing teams have well-functioning knowledge networks that allow them to share knowledge across team borders [14, 58]. Moreover, previous studies have focused on the roles of communities of practice or guilds to share tacit knowledge [59]. To complement these studies, we strengthen the focus on the role of externalized knowledge in large-scale development [16], specifically on how artifacts and trace links can be used to support inter-team coordination.

The role of team knowledge for coordination was studied in a geographically

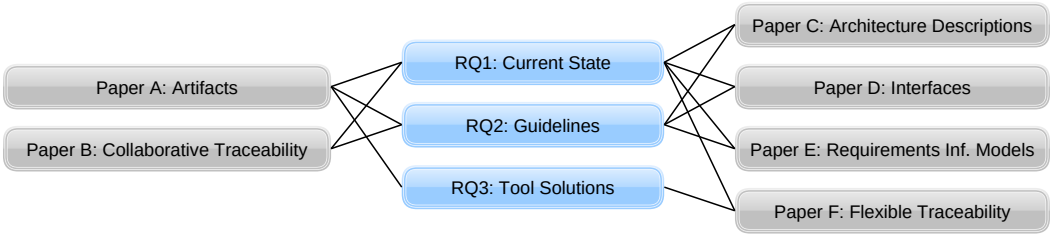

Figure 1.3: Thesis contributions and connections to main research questions

distributed telecommunications company [11]. Several technical, temporal, and process problems were found. The authors present the propositions that shared knowledge of the team and shared knowledge of the task are beneficial for coordination in software development. Similarly to this thesis, the study focuses on knowledge management for coordination in large-scale software development. Whereas the study's focus lies on one geographically distributed team, we focus on inter-team coordination between several agile teams.

Another related study focuses on artifacts in large-scale offshore software development adopting agile methods [43]. Using the data of nine international agile companies, an inventory of artifacts was created, including source code, architecture standards, test criteria, and user stories. According to the author, these artifacts act as boundary objects. Artifacts were mapped to the Scrum of Scrums development process and to agile roles. A lack of agile ceremonies was identified for the creation and refinement of architecture artifacts, risk assessment, and test plans. In this thesis, we aim to provide guidance to practitioners managing some of these artifacts in large-scale agile contexts. The focus of this thesis is not limited to software development but also investigates systems engineering concerns in large-scale agile development.

Traceability is an important enabler for stakeholders keeping track of connections between artifacts, and can also support agile methods [60,61]. To the best of our knowledge, the interplay between traceability and coordination or collaboration has been mentioned by a few previous studies (e.g., [62]), but not examined in depth. We also address the challenge of traceability management across boundaries [63] in this thesis, as well as its potential to support collaboration.

Externalized knowledge is commonly stored in tools that are used for inter-team coordination in large-scale agile contexts. For instance, the project management tool Jira or instant messaging applications can be used to support coordination [64]. It has been recommended to use application lifecycle management and product lifecycle management tool chains for large-scale agile contexts [29] or to use a lightweight, semi-automatic tool infrastructure with trace links between the product backlog, sprint backlog, tests, and other artifacts [65]. For architectural concerns, adequate tools required to manage knowledge are missing in agile methods [26]. In the automotive domain, the lack of configurable tools and the diversity of artifacts are challenges that can potentially be addressed with the development of more flexible tools and integrated tool platforms [66]. In this thesis, we contribute to a better understanding of what tooling is needed to support the management of knowledge for inter-team coordination and present tool solutions that meet practitioners' needs.

1.4 Research Approach

Software and systems engineering involves humans from several disciplines that collaborate in the creative process of developing complex products [67]. Because of the substantial role that human behavior plays in these contexts, software and systems engineering need to be studied with appropriate empirical research methods [68] that are often inspired by methods used in sociology

or psychology. The overall research approach used in this thesis has been inspired by design science [69, 70]. Starting from industry's problems and needs, we sought to gather knowledge about the state of the practice. Next, we created candidate solutions, evaluated them in cooperation with industry, and further improved the solutions. While not all of our studies were focused design science studies, they were all concerned with studying practical problems and designing contributions that aim to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development.

1.4.1 Research Focus

In this section, we elaborate on our research focus. Figure 1.4 shows an overview of the activities and contributions of this thesis. The approach shows activities of design science [69] in columns: Problem identification and design and development of solutions. In the remainder of this section, we describe the focus areas of our problem identification and solution development, as well as the evaluation methods we used.

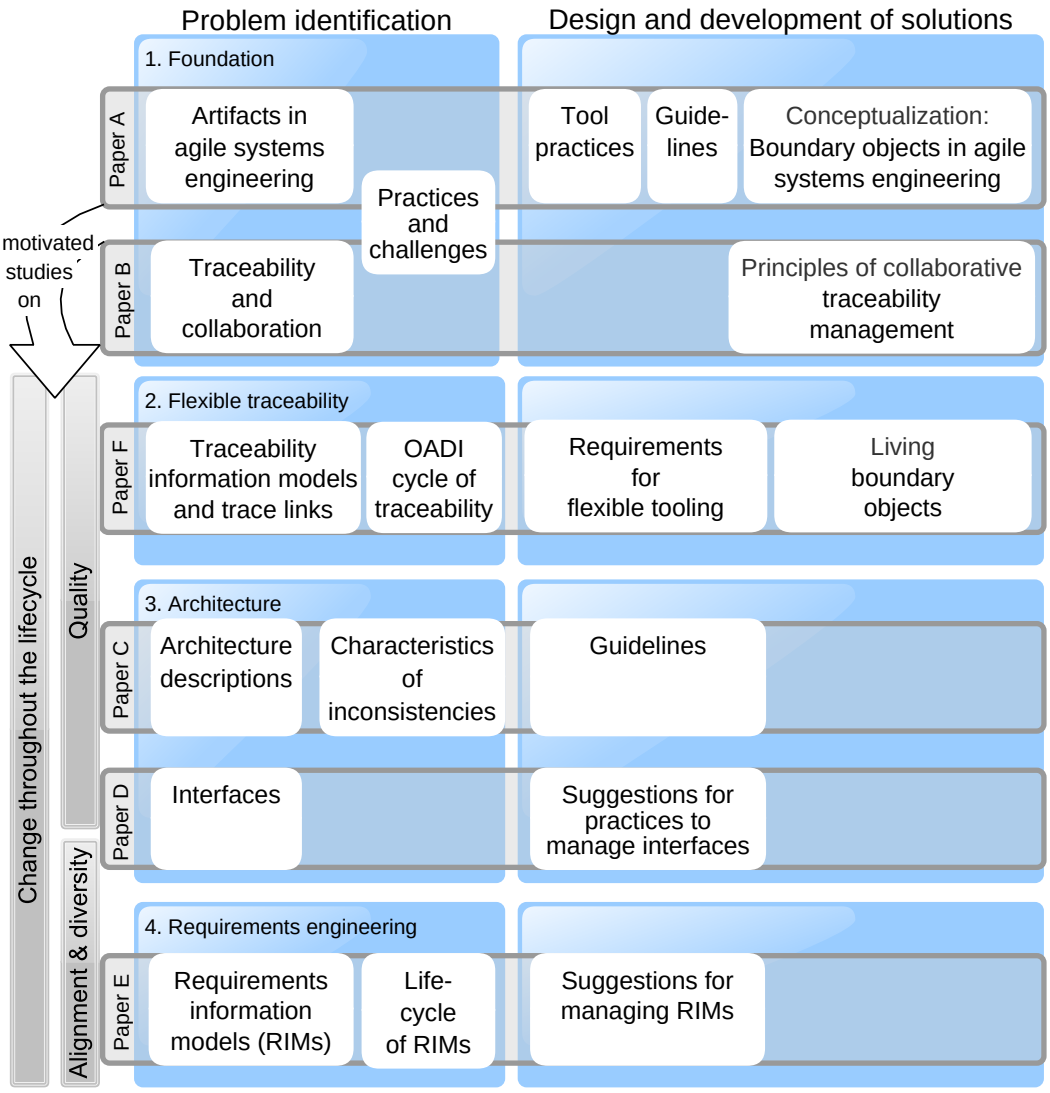

Figure 1.4: Research approach of this thesis, based on design science [69]

The rows in Figure 1.4 show Papers A–F, which were used to communicate our findings. The rows are grouped into four parts: Foundation, flexible traceability, architecture, and requirements engineering. The white boxes inside the "*problem identification*" column indicate investigated aspects of knowledge management for inter-team coordination and white boxes inside "*design and development of contributions*" show developed solutions. Papers A and B lay the foundation to motivate further studies on specific themes, as shown in gray boxes in the leftmost part of the figure. The themes are "*change throughout the lifecycle*", "*alignment and diversity*", and "*quality.*" Overall, the findings contribute to an in-depth understanding of problems, present solutions and contributions, and help to address our high-level thesis goal: to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. In the following, we briefly describe the four parts and the included contributions.

1. Foundation: Laying the foundation for our investigation of knowledge management for inter-team coordination in large-scale agile development, Papers A and B focus on artifacts in agile systems engineering, as well as traceability and collaboration. For both, we investigate practices and challenges. The created artifacts and contributions are tool practices, guidelines, a conceptualization of boundary objects in agile systems engineering, as well as principles of collaborative traceability management. The findings were evaluated using a survey (Paper A) and member checking (Paper B), as well as using subsequent studies. Papers A and B motivated further studies on change throughout the lifecycle, as we identified the need to change artifacts and trace links over time and ensure that knowledge management practices fit the specific needs of practitioners at different points in time. Moreover, the quality of trace links and artifacts was found to be important. Another aspect was the challenge of balancing alignment and diversity of knowledge management practices, i.e., how standardized or aligned practices should be and how flexibly empowered teams should select and design diverse methods in large-scale agile development.

2. Flexible traceability: This part was motivated by the identified problems and suggested contributions in Papers A and B; in particular, by the importance of distinguishing between boundary objects and locally relevant artifacts (Paper A), as well as maintaining a high level of trace link quality throughout the lifecycle (Paper B). Paper F investigates how traceability strategies can be refined over time, supported by configurable traceability information models that distinguish between different organizational scopes of trace links and artifacts (from intra-team to inter-organizational levels). We present the OADI cycle of traceability as an outcome of our problem investigation. The OADI cycle consists of the phases *Observe (O), Assess (A), Design (D), and Implement (I)* and describes how traceability strategies and concrete trace links are evolved over time. As part of the solutions, we describe requirements for flexible tooling that takes adjustable data quality levels into account and coin the concept of "*living boundary objects.*"

3. Architecture: Our analysis of artifacts in Paper A resulted in an identified need to study architecture-related artifacts, how they change over time, and how they can be kept consistent with other artifacts. We investigate architecture descriptions in Paper C, especially with a focus on inconsistencies. The contributions are guidelines that we evaluated using a survey. One of the identified types of inconsistencies was concerned with interfaces, which is what we focus on in Paper D. We investigate dimensions impacting the stability of interfaces, as well as categories of interfaces according to the dimensions. Our contributions are suggestions for practices to manage interfaces in large-scale agile development.

4. Requirements engineering: In Paper A, we identified the difficulty of supporting aligned, standardized practices, as well as diverse practices that are tailored to the needs of different teams. This challenge arises especially in large-scale agile environments in which the development of an integrated product can benefit from aligned practices, while empowered and autonomous agile teams perceive a need for diverse, team-specific practices. In Paper E, we investigate alignment and diversity in large-scale agile requirements engineering using requirements information models (RIMs). We scrutinize how RIMs can be refined throughout their lifecycles to support aligned and diverse requirements engineering practices. We contribute suggestions for managing RIMs that were evaluated using a survey.

1.4.2 Research Methods

In several studies, we investigated the current state of knowledge management for inter-team coordination in large-scale agile development and proposed solutions to improve how practitioners can manage knowledge. To this end, we selected a variety of empirical research methods. An overview of the papers with their research methods and data sources is shown in Table 1.1. In this section, we describe several potential research methods and elaborate on the reasons to select a subset of those for our empirical studies.

Controlled experiments are suitable when researchers are interested in correlations or causalities between variables. One or more independent variables are changed and effects on dependent variables are measured [68]. Experiments are appropriate when the complexity of what should be studied can be reduced to a limited number of variables and be conducted in a laboratory setting. Our research questions (Section 1.2) are description questions and design questions concerned with practical issues. We are convinced that the topic of knowledge management for inter-team coordination in large-scale agile development needs to be investigated in its actual context, considering surrounding factors of organizations, their employees, cultures, and other influencing factors. For this reason, we opted against performing a controlled experiment.

Case studies have proven beneficial when "*the boundaries between phenomenon and context are not clearly evident*" [71] and were conducted in the studies reported in Papers B, D, and F. In Paper B, we used a multiple exploratory case study, collecting data from multiple cases (i.e., 15 industrial

Table 1.1: Included papers with their research methods

Paper	Research method	Data sources	Companies
A	Design science study	11 interviewees, 17 and 25 focus group participants, 31 survey respondents, development data	6 automotive companies
B	Multiple case study	24 interviewees	15 industrial projects, several domains
C	Surveys	93 and 72 survey respondents	$\geq$ 27, several domains
D	Case study	12 interviewees	1 automotive OEM
E	Mixed methods approach	11 interviewees, 19 survey respondents, tool data and documentation	3 automotive companies, 1 tool supplier
F	Case study	6 interviewees, tool data, documentation	Focus on 1 automotive supplier

projects). Case studies allowed us to get deep insights into knowledge management for inter-team coordination by investigating a small sample of cases, considering contextual factors related to individuals and organizations.

Survey research is performed to collect data from a more representative sample in a standardized way. Using surveys, a larger amount of data can be collected more easily than when conducting individual interviews with practitioners. We used surveys in Paper C to investigate how companies deal with architecture descriptions and inconsistencies. Our design science study in Paper A and our study using a mixed methods approach in Paper E also included surveys to validate research findings.

Ethnographies can be used to study a phenomenon over time in the natural setting of a culture group. Ethnographic studies stem from sociology and anthropology and are mostly used to study human behavior in its natural context [72]. Researchers act as participant-observers and engage with the environment under study. Ethnographic studies are difficult to arrange and an appropriate context must exist. We conducted one ethnographic study in the course of the PhD studies and refer the interested reader to the publication [73] (Paper d).

Action research is concerned with observing a certain context with a problem and trying to solve the problem by introducing change using concrete actions [68]. It requires researchers to collaborate with a problem owner presenting an authentic problem. Action research is appropriate when the focus

lies on actively improving a situation and observing the effects. It would have been interesting to conduct an action research study on knowledge management for inter-team coordination in large-scale agile development. To perform an action research study, an authentic context needs to be in place, collaborators need to be willing to introduce a change in that context, and the subjectivity when observing and analyzing the effects of an action need to be mitigated.

Design science was used to inspire the overall research approach of this thesis, as well as for a focused design science study in Paper A. Design science is similar to action research in the sense that the researcher deals with an authentic context, aims to solve a problem, and evaluates solutions to the problem. However, rather than on the actions themselves and the observation of the effects, the focus lies on the creation of a design artifact in several iterations [70]. Typically, the researcher does not actively intervene and introduce change, but rather studies the context and design artifact. The environment (including people, organizations, and technology) is understood, problems and objectives are identified, and a design artifact is developed and evaluated. Several iterations of these activities can be carried out. We used the design science approach in Paper A and created practical guidelines as the design artifact.

Mixed methods research combines the potential of multiple research methods [74]. Methods can be combined in sequential or concurrent designs and can have an exploratory or explanatory focus. While mixed methods research gives researchers a more complete picture of the phenomenon under study, it is challenging to extensively collect, analyze, and triangulate the required data [68, 75]. We used a sequential design in Paper E, making use of data from interviews, surveys, tool data, and documentation.

As described in this section, we use a variety of research methods in this thesis, including quantitative and qualitative ones. The selection of methods indicates that diverse skills and methodical knowledge to conduct research have been acquired in the course of this PhD project.

The findings of the included publications contributed to our overall research goal and main findings of this thesis. To synthesize our findings, we followed a thematic synthesis approach [76]. Thematic analysis can be used to identify and analyze research findings from different studies and create higher-order themes to report on. In our thematic analysis, we extracted data from Papers A–F to summarize their aims, contexts, and findings (see Section 1.5). The data was coded with an editing approach, starting with a priori codes that reflected the thesis' research questions ("*current state*", "*guidelines*", and "*tool solutions*"). Codes were added, refined, grouped, and reviewed to arrive at themes. On paper, we created a mind map of themes, indicated relationships between them, and ensured that we kept track of which studies contributed to which themes. We further analyzed themes to arrive at the higher-order themes and main findings reported in Section 1.6. Thematic synthesis is a flexible method and works with heterogeneous evidence types, but its lack of transparency is a potentially challenging issue [77]. To improve transparency, we aimed to establish a chain of evidence by indicating how research papers contributed to the respective themes in Section 1.6.

1.4.3 Threats to Validity

The research approach we used in this thesis possesses several strengths and weaknesses. A strength is that the underlying studies were conducted in close collaboration with industry and are relevant for practitioners. They allowed us to study knowledge management for inter-team coordination in depth and in the specific contexts of companies. Based on a foundation focusing on artifacts, traceability, and collaboration, we further investigated areas we identified as important in the context of knowledge management for inter-team coordination in large-scale agile development, i.e., flexible traceability, architectural knowledge, and requirements engineering. The developed solutions in Papers A, B, C, and E were evaluated using surveys or other member checking techniques. For Papers D and F, the suggested solutions have only been informally evaluated with collaborating companies. Moreover, further studies are needed to evaluate tool practices and requirements for flexible tooling in realistic contexts.

A potential risk to this industrial PhD project is concerned with the role and influence of Systemite AB, the collaborating company. Being a tool provider of the systems engineering tool SystemWeaver which is used in large-scale agile companies, Systemite has been interested in the performed studies and their findings. From the beginning, stakeholders in the company articulated this interest, but stressed that they did not intend to influence the research agenda. This decision was facilitated by the fact that it was not the company funding the PhD project, but a foundation. Moreover, we did not study practices at Systemite AB, but rather at customers or other collaborating companies. The employment was set up in a way that 80% of the time was reserved for research activities. On average, at least three days a week were spent at Chalmers University of Technology. We were involved in a dialog regarding the research goals together with academic and industrial supervisors and the examiner. Throughout the course of this PhD project, it was ensured that the publication of findings could happen in an independent way and without any delay enforced by the company. The close collaboration with academic researchers helped to mitigate potential issues.

With the presented strengths and weaknesses in mind, we now present threats to validity. Detailed threats to validity are described in the respective sections of our papers. Several of our studies examine industrial cases in depth with qualitative approaches, but we also use quantitative approaches as complementary methods. We discuss threats to validity for case study research [67], but also threats that arise when combining qualitative and quantitative methods [75].

Internal Validity: Internal validity refers to causal relations and potentially influencing factors that we were unaware of, but influenced the findings. For instance, in Paper E, we collected factors motivating the need for alignment and diversity and could have potentially missed relevant factors. To mitigate threats to internal validity or credibility, we aimed to provide contextual information and authentically report on the findings using rich descriptions [75]. Posing open questions in an exploratory fashion helped us to investigate knowledge management for inter-team coordination from a broad perspective. More-

over, we used follow-up questions in interviews and focus groups to better understand participants' explanations and potential confounding factors. In our surveys, we were limited to predefined questions, but added comment fields to capture additional aspects. Using peer debriefing was another mechanism of making sure that we discussed influencing factors [67].

Furthermore, we mitigated threats to internal validity by triangulating data and combining quantitative and qualitative methods. We used several methods in our design science study (Paper A) and in our mixed methods study (Paper E), which allowed us to gather rich data from a variety of sources and validate our findings.

Construct Validity: This aspect of validity refers to whether our measures are adequate to capture the concepts, theories, and constructs we intend to study [67]. When investigating how knowledge is managed for inter-team coordination in large-scale agile development, many terms are used that can be understood in various ways, e.g., *artifacts* or *traceability*. Whenever several domains and disciplines are involved, it can be challenging to ensure that a common understanding of such terms is created. We aimed to mitigate this threat using an initial introduction of relevant terms in the interviews we conducted and by starting surveys with short explanations of basic terms and concepts. Being employed as an industrial PhD student at a tool supplier company, we could gain in-depth insights into practices and typical terms used for concepts in industry in a prolonged engagement [67].

External Validity: External validity is concerned with the generalizability of our findings and the degree to which our findings are transferable to other companies, individuals, and situations than the ones studied in this thesis [67]. Our case studies focus on the in-depth and thick descriptions of individual cases, rather than on having broad generalizability [78]. Companies that are located in different places, have employees with different backgrounds, or use different tools might have different experiences and challenges. For some studies, external validity is higher, as more companies and individuals have been involved. Our exploratory study in Paper B was conducted together with 24 individuals from 15 industrial projects in two countries and gives indications of how traceability management is generally conducted. In the surveys reported in Paper C, 93 and 72 participants from different domains participated.

To improve transferability, we carefully described the characteristics and contextual factors of our participating companies and projects [78]. These descriptions can help readers to reflect on whether the findings could be valid also for other cases. Some of our studies were conducted with automotive companies, a domain that comes with particularly challenging characteristics (e.g., due to the variety of involved disciplines, heterogeneous functions, important quality attributes, and OEM-supplier relationships [79]). We cannot claim transferability of those findings to other domains, but expect that large-scale agile contexts with less challenging characteristics can make use of those contributions and adjust them to their needs.

Reliability: Reliability is concerned with whether the findings, data, and analyses are consistent with what should be measured or whether they de-

pend on specific researchers. To improve the replicability of our studies, we made our instruments available (e.g., interview guides, survey questions, and documentation of our analysis methods). Moreover, we aimed for high transparency and consistency regarding the chain of evidence of our research findings. We use quotes and state the roles of participants to make the deduction of research findings more transparent. Member checking helped us to check whether we had correctly understood our participants' statements and validate our research findings.

While we aimed to describe our research methods in detail, the analysis of qualitative data using coding was one of the activities that are researcher-dependent. We illustrated our coding approach by giving examples and providing our analysis guides in Papers A and B.

Peer debriefing was another mechanism to improve reliability and make sure that our methods were less dependent on individual researchers. The input gathered from peer debriefing allowed us to phrase statements in a more unambiguous way.

1.5 Contributions of the Thesis

Our contributions are included in the six papers attached to this thesis. As described in Section 1.4, our overall research approach was based on design science: we identified problems, designed and developed solutions, and evaluated them. In the following, we present the underlying research questions and findings of the papers and stress the papers' relations to the overall goal: to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development.

1.5.1 Paper A: Artifacts in Agile Systems Engineering

In Paper A, we lay the foundation for subsequent studies by analyzing how knowledge is currently managed and providing initial solutions to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. As more and more large-scale companies adopt agile methods, they aim to reduce unnecessary documentation and optimize knowledge management practices [80, 81]. Explicit knowledge is commonly manifested in artifacts. Some artifacts are used to externalize knowledge for inter-team coordination, whereas others are created as a by-product of development activities of an individual or a smaller team. Focusing on artifacts and their roles for inter-team coordination in this paper helped us understand current practices and give guidance to practitioners managing knowledge in large-scale agile development.

Related work proclaimed the need for more research to give guidance to practitioners on what artifacts are needed and how to manage them in large-scale agile development [17]. We collaborated with six automotive companies to scrutinize how artifacts are managed in practice, what challenges occur, and how practitioners could be supported by guidelines and tooling. The study's research questions and contributions are:

RQA-1: What are practices to manage artifacts in agile automotive systems engineering?
RQA-2: What practical challenges exist with managing systems engineering artifacts in agile automotive contexts?
RQA-3: How can the management of systems engineering artifacts in agile automotive contexts be supported by a systems engineering tool?
Guidelines: Guidelines to manage artifacts in agile systems engineering were the design artifacts we created using our design science approach.

Answering RQA-1, we developed an inventory of artifacts in agile automotive systems engineering. In automotive companies, systems' lifecycles are typically long and knowledge needs to be permanently retained. In this paper, we distinguish between boundary objects (used for inter-team coordination and to create a shared understanding across team borders) and locally relevant artifacts (used within a team).

Some of the identified artifacts are candidates for boundary objects: architecture models and descriptions, high-level requirements, and variability information. These artifacts are often spread throughout several tools and systems, represented in several formats, and created at different points in time.

Moreover, we identified locally relevant artifacts that are managed within the scope of an agile team, i.e., documentation, low-level requirements and tests, Simulink models, and signal databases. Locally relevant artifacts are created, maintained, and used within an agile team. Documentation, low-level requirements, and tests are created by team members in text form or code. Simulink models and signal databases are prescriptive artifacts and required to be consistent so that the developed system works as intended.

We found that our participants face challenges with the trade-off between diversity and alignment of teams, degradation of artifacts, a mix of plan-driven and agile methods, deciding what artifacts one should regard as important, high staff turnover, and different locations and backgrounds (RQA-2).

We suggested several practices to manage artifacts in a systems engineering tool (RQA-3). To identify boundary objects, trace links between artifacts can be analyzed: artifacts traced to by artifacts owned by different teams are candidates for boundary objects. High trace link quality is essential to support this way of identifying boundary objects. Boundary objects should be managed with care, as they play a crucial role for inter-team coordination. They should be kept recognizable for users, which can be supported by visualization features. Moreover, change management mechanisms for boundary objects are beneficial and it is recommended to keep boundary objects as stable as possible. For locally relevant artifacts, we suggest managing them with lightweight processes, making them reusable, and leveraging data export and code generation features.

Our guidelines recommend identifying artifacts, deciding on a strategy on a high level, and evaluating artifacts' relevance and usage at frequent intervals. Coordination mechanisms around boundary objects should involve representatives from different teams who create initial boundary objects upfront with a lightweight approach. Locally relevant artifacts should be produced as late as possible and be made reusable. Traceability to boundary objects is crucial to support change propagation.

Following these guidelines, practitioners can distinguish between knowledge that is central for stakeholders from different teams (i.e., boundary objects) and knowledge that can be flexibly managed within a team. With bespoke approaches and activities to manage these types of knowledge, inter-team coordination can be more easily supported and empowered teams can manage intra-team knowledge in a flexible and agile way.

1.5.2 Paper B: Collaborative Traceability Management

Traceability is an important enabler of knowledge management and supports several knowledge management activities, as we described in Section 1.1.4. A recognized challenge with traceability management is to deal with large-scale development contexts involving multiple stakeholders that need to collaborate [63]. In Paper B, we focus on the interplay between collaboration and traceability, so that both can be improved to support knowledge management for inter-team coordination in large-scale agile development. We approach the topic from the angles of organization, process, and culture, and analyze how characteristics of the development effort influence traceability management and collaboration.

Concretely, we focus on the following research questions:

RQB-1: What are practitioners' challenges with collaboration in traceability management?
RQB-2: How can traceability management support collaboration?
RQB-3: How does collaboration relate to different approaches of traceability management?
RQB-4: What characteristics of the development effort influence traceability management and collaboration?

To collect data, we conducted 24 semi-structured interviews with practitioners from 15 industrial cases in Germany and Sweden. We categorized identified challenges (RQB-1) into those related to collaboration across tool or organizational boundaries, those related to common goals and responsibilities, and challenges related to trace link maintenance. The latter category includes issues concerned with trace link quality, as well as change propagation and notification. In situations in which these challenges occur, traceability and collaboration typically exacerbate each other.

We also identified opportunities and ways to support collaboration with traceability management (RQB-2). Traceability management can facilitate communication in a distributed environment, support inter-disciplinary engineering, provide mechanisms to explicitly document decisions, and establish new incentives for stakeholders that create trace links to receive information.

Moreover, we identified three approaches to traceability management that address collaboration in different ways (RQB-3). Requirements-centered traceability management is common whenever formal approaches to collaboration are needed and customer-supplier relationships play a role. Developer-driven traceability management focuses on tracking issues or tickets and commits. Collaboration is typically more informal and occurs in smaller agile teams.

Mixed approaches combine requirements-centered and developer-driven traceability management and rely on formal and informal collaboration.

We further analyzed what characteristics of the development effort influence traceability management and collaboration (RQB-4). These characteristics are most prominently observable in the underlying development paradigms and the rigor of following them. With rigor, we refer to how consequently a traceability strategy is defined and followed. Rigorous cases systematically establish and maintain traceability, independently of whether they follow agile or plan-driven paradigms. Less rigorous cases report more challenges, especially with respect to trace link quality, but also with respect to collaboration across boundaries.

This paper proposes four principles of collaborative traceability management: First, to put stakeholders' information needs and goals of traceability at the center and select lightweight, developer-driven or formal, requirements-centered approaches, depending on the identified needs. The second principle is to balance the effort and benefit of traceability management per role. Third, practitioners require mechanisms to enable change propagation and notification across boundaries. The fourth principle is to strive for a rigorous culture with respect to traceability maintenance. These principles support managing traceability-related knowledge, so that coordination between teams can be facilitated. Following these principles is crucial to support knowledge management for inter-team coordination, especially in large-scale development. In Paper F, we stress that living boundary objects should be traced to and from other artifacts, which can be supported by the proposed principles of collaborative traceability management.

1.5.3 Paper C: Architecture Descriptions

Architecture models and descriptions can potentially serve as boundary objects, as we identified in Paper A. Paper C aims to study architecture descriptions in further detail. In agile environments, the architecture of systems evolves over time [82] and architectural concerns need to be coordinated between agile teams. In Paper C, we focus especially on architectural knowledge for inter-team coordination in large-scale agile development, how it is typically managed, and how practices can be improved. In practice, not all architecture descriptions actually serve as living boundary objects and are used by agile teams as intended by their producers [83]. As a consequence, inconsistencies emerge between architecture descriptions and code, as well as between multiple architecture descriptions. However, more empirical research was needed to investigate how inconsistencies arise [84]. To this end, our research questions are:

RQC-1: What are the types, reasons, and consequences of architectural inconsistencies?
RQC-2: How do architecture descriptions and their relation to the implementation change over time?
RQC-3: What are guidelines to support practitioners with the management of architecture descriptions?

We conducted two surveys to answer our research questions. With respect to RQC-1, we found that inconsistencies typically arise due to broken rules, constraints, patterns, guidelines, inconsistent wording and language, or interface specifications not being followed in the implementation. According to our respondents, there are several reasons for inconsistencies, e.g., lack of time, lack of knowledge, wrong assumptions, or lack of communication. The reported reasons point to the issue that architecture descriptions are not considered useful by all stakeholders. Consequences of inconsistencies are typically observable late in the lifecycle, e.g., during maintenance.

We found that architecture descriptions and their relation to the implementation change over time (RQC-2). Our respondents stated that an initial description of the architecture should be refined with emerging elements during development and design. Consistency becomes a more central issue during development and design. Whereas some respondents stated that the architecture description should describe the current state of a system or software, others reported that the future architecture should be described. Both time perspectives appear relevant, but should not be mixed.

We presented six guidelines (RQC-3) to manage architecture descriptions:

[a] State the purpose and audience to define the level of abstraction and inclusion criteria.

[b] Distinguish between the present and future perspectives.

[c] Minimize the number of elements in the upfront architecture and include elements that are relevant across team boundaries.

[d] Assess decisions in the description of the future architecture in communities of practice.

[e] Integrate architects into the teams to understand emerging aspects.

[f] Use one source of information and make it traceable.

With these guidelines, architecture descriptions can be established as living boundary objects that are kept up to date and support coordination between teams. When managed appropriately, architecture descriptions can thus contribute to supporting knowledge management for inter-team coordination.

1.5.4 Paper D: Interfaces for Agile Architecting

Interfaces are important elements in architecture descriptions and their coordination is key for successful agile architecting [80]. Practitioners need appropriate solutions to manage interface-related knowledge and support coordination between teams. Interface-related knowledge affects how development teams coordinate activities, perform integration of components and subsystems, and gain a shared understanding of the system to be developed. Agile teams need interfaces that constitute "*islands of stability*" [26], which enable them to autonomously work on their parts of a system. Paper D focuses on how interface-related knowledge is managed to support coordination between teams. We aim to investigate how interfaces are changed over time and what dimensions impact stability. Our research questions are as follows:

RQD-1: What dimensions impact how interfaces in agile automotive contexts are changed and how are the dimensions related?
RQD-2: What categories of interfaces exist in the context of agile systems engineering with respect to the dimensions?
RQD-3: To what degree does an interface's category change over time in agile automotive contexts?

Based on semi-structured interviews with employees of an automotive OEM, we derived eight dimensions impacting how interfaces are changed (RQD-1). Stability of interfaces is a central dimension of interfaces and typically, the time to perform a change leads to interfaces being kept stable. The levels of abstraction and criticality also typically imply higher stability of interfaces. Moreover, changes take more time the more components use an interface, as the distance to affected parties is typically higher and coordination requires more time and effort. Also, the maturity of affected functions and the position in an interface's lifecycle generally increase an interface's stability.

We derived three categories of interfaces based on the dimensions (RQD-2). Commodity interfaces are used between mature components and are characterized by high stability and a long time to perform a change. On the other extreme, there exist early stage interfaces that are still under prototype development and between teams with a short distance. Finally, we investigated central vehicle interfaces, that are very stable and affect teams across a large distance. Due to the high criticality and number of affected components, they are difficult to change and require strict coordination mechanisms.

Typically, dimensions rather increase than decrease (RQD-3)—and interfaces become more stable, used by more components, become more critical, and take more time to change. Interfaces typically reach a "*point of increased stability*", where change becomes more complicated.

We present five suggestions for practices to manage interfaces and support coordination between agile teams. In total, they contributed to our overall goal of supporting knowledge management for inter-team coordination. We recommend selecting what interfaces need to be managed centrally and leave early stage interfaces up to the teams. As soon as the lack of stability impedes that teams can work autonomously, stakeholders with architecture expertise should be involved to establish interfaces that set the boundaries. Interfaces should be kept as abstract and generally applicable as possible. Moreover, we suggest assessing them in smaller groups in early stages. Central vehicle interfaces require controlled change mechanisms and a strategy for versioning, as they typically affect multiple teams whose concerns need to be coordinated whenever a change should be performed.

1.5.5 Paper E: Alignment and Diversity of Requirements Engineering Practices

In large-scale organizations, several stakeholders and groups work with requirements engineering and need to weave the consolidated requirements into a "*coherent story*" [85]. Knowledge needs to be managed to support the establishment of requirements engineering practices and inter-team coordination around them. Attempts have been made to create company-wide methods for

requirements engineering and align practices [86], but also to support diversity and customized practices [73, 87]. Aligned requirements engineering practices facilitate the establishment of general, centrally organized coordination mechanisms, whereas diverse practices are required to meet the needs of interdisciplinary and heterogeneous teams. Requirements information models (RIMs) represent what entity types of information with what relationships and constraints should be created to capture requirements-related knowledge. RIMs are used to specify what knowledge is captured, which in turn facilitates inter-team coordination around requirements engineering practices. Our goal is to analyze how the balance of alignment and diversity is and can be supported by RIMs, as captured in the following research questions:

RQE-1: What are reasons to balance alignment and diversity of RE practices in large-scale automotive companies?
RQE-2: How do RIMs enable the balance of alignment and diversity of RE practices in large-scale automotive companies?
RQE-3: What actions can be observed when large-scale automotive companies balance alignment and diversity using their RIMs?
RQE-4: What are suggestions for managing RIMs to balance alignment and diversity of RE practices?

We found that alignment is needed to facilitate integration, establish a common language, increase the quality of requirements, and adhere to standards (RQE-1). Diversity is required to support the needs of various disciplines, different methods, natures of functions, and elicitation practices.

RIMs support alignment by allowing to specify common entity types, relationships, attributes, consistency checks, maturity levels, and Definition of Done criteria (RQE-2). On the other hand, diversity is enabled by supporting generic relationships, the creation of subtypes, free text fields, and several ways of organizing backlogs and projects.

With respect to actions to balance alignment and diversity (RQE-3), we observed that practitioners carefully relate the lifecycle of the RIM and the lifecycles of concrete requirements instantiations. The lifecycle of concrete requirements requires diversity in early phases, but alignment especially as the product is released. Alignment is typically ensured by adding consistency checks, whereas practitioners support diversity by evolving the RIM based on observed needs when working with concrete requirements.

We suggest involving key stakeholders, establishing a common and aligned structure on a high level and diversity on a lower level, and evaluating changes to a RIM with a small group of users (RQE-4). Separate requirement types or attributes are beneficial for requirements with special procedures. It should be possible to create minimal information first and fill in details later on. The RIM should be kept as general as possible and stakeholders should rely on training and communication rather than on too strong restrictions. Inter-team coordination can be facilitated if a RIM is managed with sufficiently aligned and diverse practices. RIMs create a shared understanding of requirements engineering practices that can be leveraged to coordinate activities around integration, standards, or quality assurance. At the same time, a balance between alignment and diversity allows agile teams to be empowered and use methods that support their individual needs.

1.5.6 Paper F: Flexible Traceability

Traceability is a potential facilitator of knowledge management for inter-team coordination, as trace links capture important knowledge about how artifacts from different organizational groups are related. Many existing traceability management approaches neglect the fact that practitioners' needs change over time and depend on the organizational groups involved in managing particular trace links. Traceability information models (TIMs), as well as other parts of the traceability strategy, have to fulfill stakeholders' needs [88]. In this paper, we present how traceability can be evolved over time to support inter-team coordination and stakeholders' changing needs. Concretely, we focus on

F-1: what characteristics trace links with different organizational scopes have—from intra-team to inter-organizational trace links,

F-2: how traceability strategies and practices were established and tailored over time at an automotive supplier, and

F-3: requirements to create flexible tool solutions that support evolving traceability needs.

Over the course of three years, we followed the development of the traceability strategy and practices at the automotive supplier Zenuity. We found a crucial difference between trace links used within a team and trace links that are used across team borders. Lightweight coordination approaches were leveraged for trace links between artifacts created and used in the same team, whereas more rigid and formal approaches were used for larger organizational distances (F-1). For instance, intra-team links could be flexibly changed in face-to-face meetings, whereas links to customer requirements were formally versioned and kept as stable as possible. Trace links between artifacts on the same level of abstraction (e.g., between two components) were managed more flexibly than trace links connecting artifacts on different levels of abstraction (e.g., trace links from design artifacts to functional requirements).

We describe the OADI cycle of traceability (F-2) to explain how traceability was established and tailored over time, starting with observing (O) and assessing (A) current practices and practitioners' needs. In later iterations, quality checks were used to assess how the TIM was instantiated and used. As technological or business changes were performed, artifacts evolved and potentially became boundary objects. For instance, at some point, a redesign of the architecture resulted in intra-team signals becoming boundary objects. The gathered information was used to design (D) and implement (I) changes to the traceability strategy. For example, the TIM was redesigned to improve decoupling between artifacts. Users received training, coordination mechanisms were adjusted, and acceptance criteria were tailored. While artifacts and trace links for safety analysis did not have to be complete in early stages, this aspect was soon included in the acceptance criteria. We describe the distinct role of "*living boundary objects*", that are kept up to date, traced to other artifacts, and serve for inter-team coordination. These boundary objects helped to support change and maintain a common understanding across sites, even as the traceability strategy was adjusted.

Tool support needs to be flexible to consider the fact that practitioners'

needs evolve and depend on the involved organizational groups. Existing tools support flexibility to some extent, e.g., by providing a configurable TIM. We present requirements for flexible tool solutions (F-3). Stakeholders should be supported in determining adequate data quality levels for specific trace links and artifacts at specific points in time. For instance, a tool could automatically suggest adding more quality checks for an artifact that recently became a boundary object. Moreover, adaptable versioning solutions can be beneficial for artifacts and trace links. Loose, coarse-granular versioning might be appropriate for intra-team links at early stages, whereas strict versioning should be recommended for boundary objects and record explicit information of all changes. Organizational traceability should be supported to keep track of links between stakeholders with their roles, groups, responsibilities, and artifacts in use. For all artifacts and trace links, it is beneficial to record stakeholders that are potentially affected by a change. Moreover, the detection of change, change propagation, and trigger tuning should be supported by a tool. For instance, if the TIM has been changed, instance data should be evolved. Other changes might affect the desired level of quality or versioning for certain artifacts or trace links. Living boundary objects can be powerful facilitators of change and create a common understanding across team borders, even though traceability strategies and practices are adjusted over time. The suggested insights can help to establish knowledge management and coordination mechanisms that can be evolved with practitioners' needs.

1.6 Answering the Thesis' Research Questions

The previous sections have introduced the six appended papers and their main contributions. Our overall goal is to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. We aimed to achieve this goal by analyzing the current state of managing knowledge for inter-team coordination and designing solutions. The papers contributed in several ways to the research questions of this thesis. Our questions are concerned with the current state of managing knowledge (RQ1), but also with guidelines (RQ2) and tool solutions (RQ3) to manage knowledge for inter-team coordination. As stated before, we mainly focus on externalized knowledge, manifested in artifacts or trace links, and used for development activities.

1.6.1 RQ1: Current State of Managing Knowledge

In the following, we answer **RQ1**: *What is the current state of managing knowledge for inter-team coordination in large-scale agile development?*

All papers contributed to answering this question, with several main findings that all emerged from at least one paper. The main findings are summarized and concisely stated in boxes in this section. Figure 1.5 shows an overview of the findings and what papers we used as the basis to explore them. Findings are depicted as blue boxes and papers are shown in gray. A paper contributes to a finding if an arrow connects the paper's and the finding's box. All papers contribute to our analysis of *heterogeneous characteristics* and

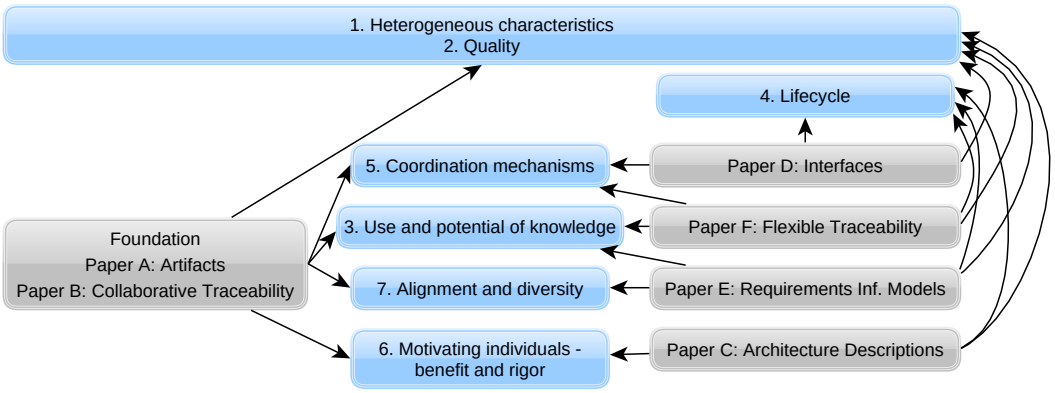

Figure 1.5: Overview of findings related to the current state of managing knowledge (RQ1)

quality (findings 1 and 2). Papers A and B lay the foundation of our investigation of artifacts and collaborative traceability management, whereas Papers C–F present particular cases of knowledge management in the *lifecycle* of artifacts and trace links (finding 4). We further investigated the *use and potential of knowledge* (3), as well as *coordination mechanisms* (5) around knowledge management. We found that to *motivate individuals* to invest in knowledge management (6), the benefit of knowledge management is stressed or rigor is enforced. Finally, we investigated the balance of *alignment and diversity* of knowledge management practices (7).

1. Heterogeneous characteristics of externalized knowledge: This finding is concerned with the heterogeneous characteristics of externalized knowledge. In the study in Paper A, we found that knowledge is manifested in a variety of artifacts of different formats, on different levels of abstraction, stored in various tools, some being descriptive and some prescriptive, and used by different organizational groups and stakeholders. We identified that some artifacts serve as boundary objects and some are locally relevant artifacts, being created, maintained, and used within a team. We investigated several boundary objects in depth. With respect to architecture-related knowledge, architecture descriptions were in the focus of Paper C and interfaces of Paper D. We found that architecture descriptions can have heterogeneous characteristics and describe the current or the future state of an architecture (Paper C). Interfaces can have different levels of abstraction, criticality, and are used by varying numbers of components by teams of different organizational distances (Paper D). Requirements information models (RIMs) can serve as boundary objects (Paper E) and define how requirements can be created and traced. Potentially, several RIMs with different characteristics exist in large-scale organizations. Traceability is needed to keep track of connections between artifacts and managed with requirements-centered, developer-driven, or mixed approaches (Paper B). Trace links can be on various levels of granularity, have types, versions, and can be changed with different mechanisms in different tools (Papers B and F).

We capture these findings from our papers in this thesis' first main finding:

Main finding 1: Heterogeneous characteristics

Externalized knowledge in large-scale agile development can take various forms and is typically manifested in multiple heterogeneous artifacts and trace links.

2. Quality of externalized knowledge: To be useful for practitioners, knowledge needs to be reliable and of sufficient quality. Quality issues related to artifacts and trace links were an emerging theme in several studies, e.g., Paper A. We focus on several quality attributes in this thesis:

Up-to-dateness is one of the attributes that we describe as an underlying concern in all papers. It is challenging to keep artifacts and trace links up to date and companies face the challenge of artifact degradation (Paper A). This challenge motivates the establishment of change management mechanisms that ensure that artifacts and trace links are updated as necessary.

Consistency is concerned with whether artifacts representing the same concepts are in agreement with each other (Paper F). Inconsistencies among architecture descriptions and between architecture descriptions and code are investigated in Paper C. Architectural inconsistencies arise, for instance, when interface specifications are not followed in the implementation, but also when wording conventions or rules are disregarded. Inconsistencies have severe consequences during testing, maintenance, deployment, or at runtime. Consistency checks for requirements were mentioned in Paper E as a way to enable alignment.

Version consistency is important when a tool supports the creation of versions for artifacts and trace links. Version consistency is achieved when trace links and artifacts have the intended versions (Paper F). For instance, version inconsistencies arise when one component uses an outdated version of an interface.

Completeness was found to be crucial in Paper B. In the context of traceability, it means that the set of trace links should be complete. Stakeholders can have difficulties leveraging the use of traceability if trace links are missing.

Validity relates to the conformity of artifacts and trace links to the metamodel or information model (Paper F). Requirements or traceability information models (RIMs or TIMs) can be adjusted with adequate tooling, which raises the need to adjust instance data so that it corresponds to the changed information models.

Main finding 2: Quality

Quality of artifacts and trace links is required to leverage the benefit of externalized knowledge. We identified several relevant quality attributes, i.e., up-to-dateness, consistency, version consistency, completeness, and validity.

3. Use and potential of externalized knowledge: As part of the knowledge management cycle, knowledge should be acquired and applied (Section 1.1.3). We found several ways in which externalized knowledge can be leveraged to

support software and systems engineering activities, many of which relate to inter-team coordination. For instance, signals can be used as horizontal boundary objects to establish a common understanding between component teams and facilitate inter-team coordination (Papers A and F). RIMs can be used to balance alignment and diversity of requirements engineering practices (Paper E). Trace links can facilitate several activities, e.g., progress tracking or change impact analysis. Traceability also has the potential to facilitate communication, support inter-disciplinary engineering and explicit documentation of decisions, and motivate stakeholders to invest in traceability to receive notifications (Paper B).

Often, the use and potential of externalized knowledge cannot be fully realized due to the invisible benefit of traceability (Paper B). The creators of trace links are typically not their primary users and do not invest in sufficient quality. This point strengthens the importance of motivating individuals to manage knowledge (see finding 6).

Main finding 3: Use and potential of externalized knowledge

Externalized knowledge can be used to support various activities, many of which relate to inter-team coordination. For instance, boundary objects can be used to create a common understanding of a system, requirements information models can establish sufficiently aligned requirements engineering practices to facilitate integration, and trace links can support change impact analysis and progress tracking.

4. Lifecycle: We found that knowledge is typically manifested in artifacts and trace links that undergo periods of change and stability (Papers C, D, and E). It is important to keep artifacts and trace links up to date (finding 2), which is why change notification and propagation need to be supported. Some boundary objects are designed early on and used to create other artifacts, e.g., architecture descriptions or RIMs (Papers C and E). They define high-level rules and constraints that enable inter-team coordination. During development and design, emerging elements are captured and boundary objects are further refined. Needs observed on a concrete (instance) level trigger changes on an abstract level (Papers E and F). While diverse ways of modeling RIMs were found to be more important in early phases, aligned and consolidated practices play a central role later on, before artifacts might be deprecated. For interfaces, we found that several dimensions (e.g., the number of components using an interface, the level of abstraction, criticality, and maturity of related functions) increase the time to change an interface and the enforced stability (Paper D). Similarly to RIMs, TIMs are tailored and adapted, which can be explained using the OADI cycle of traceability (Paper F), consisting of the phases *Observe (O), Analyze (A), Design (D),* and *Implement (I).* Flexible mechanisms are needed to ensure that stakeholders' needs are met in an evolving environment.

Main finding 4: Lifecycle

Externalized knowledge goes through phases of change and stability over time. During early development, dedicated boundary objects are used to describe high-level constraints to support inter-team coordination and the creation of other artifacts. Examples of such boundary objects are architecture descriptions, RIMs, or TIMs. Based on experiences with concrete artifacts, boundary objects are then refined, become more stable, and lead to more aligned practices. The OADI cycle of traceability and the lifecycle of RIMs illustrate how externalized knowledge can be adapted over time to meet stakeholders' needs.

5. Coordination mechanisms: In traditional agile methods, face-to-face communication is commonly used to coordinate and exchange knowledge. In large-scale environments, practitioners face challenges when collaborating across organizational boundaries (Paper B). The variety of backgrounds, disciplines, methods, and locations raises the need for adequate coordination mechanisms for quickly-changing contexts. Boundary objects are one way of supporting coordination and establishing a common understanding between teams. They often arise at team or inter-organizational borders and are ideally kept as living artifacts (Paper A). Supporting coordination mechanisms change depending on the scope of artifacts and trace links (Paper F): rigid and formal mechanisms are used for trace links to external artifacts (e.g., customer requirements), less formal mechanisms are used for boundary objects inside an organization, and intra-team trace links are managed with a very lightweight approach. Requirements-centered traceability is managed in a more formal way than developer-driven traceability, for which face-to-face communication is commonly used (Paper B). In Paper D, we report that the higher the number of affected components and the higher the organizational distance between teams, the longer changes take, as the coordination between involved stakeholders takes more time.

Main finding 5: Coordination mechanisms

A variety of coordination mechanisms are used when managing knowledge in large-scale agile contexts. Informal approaches can include face-to-face communication between teams and the lightweight establishment of boundary objects, whereas coordination mechanisms between different organizations include strict documentation of changes. Generally, we found that the larger the distance between teams, the more rigorous and formal are the coordination mechanisms.

6. Motivating individuals to manage knowledge—benefit and rigor: Humans can be intrinsically motivated to manage knowledge because they see the benefit, but also extrinsically motivated because practices are followed with a certain level of rigor. Externalized knowledge can be leveraged for several activities (see finding 3). For traceability, the benefit and potential

are often invisible to the stakeholders creating trace links, since the creators are typically not the main users. In practice, the quality often suffers because trace links are not maintained, and the use of traceability cannot be completely realized. On the other hand, we found that if traceability management is successfully conducted, it can also positively influence collaborative activities (Paper B). The reasons for architectural inconsistencies reported in Paper C are, among others, a lack of time, lack of knowledge, and lack of communication. These reasons indicate that counteracting inconsistencies is not prioritized, as practitioners do not see the benefit in keeping architecture descriptions and implementation artifacts consistent.

We found that a certain level of rigor is perceived as necessary to ensure that trace link quality is on an appropriate level to utilize the potential of traceability (Paper B). A rigorous culture requires the development paradigm and practices to be defined, communicated, and followed in an organization. In our study on traceability management, cases that had a low level of rigor reported more severe challenges with quality, independently of the development paradigm.

Main finding 6. Motivating individuals—benefit and rigor

Individuals need to be motivated to manage knowledge and support inter-team coordination. Motivation can be improved by indicating the benefit of knowledge management or by establishing rigorous practices. If neither the benefit is clear nor rigorous practices are enforced, artifacts and trace links are not maintained and quality suffers.

7. Alignment and diversity: The trade-off between alignment and diversity was identified as a challenge with managing artifacts in agile systems engineering (Paper A). After all, a common, integrated product needs to be developed and it helps to rely on standardized or aligned methods, in particular, when managing knowledge for inter-team coordination. At the same time, teams should be empowered to make their own decisions and work with tailored methods. We phrased this point as the trade-off or balance between alignment and diversity. As this issue is especially observable in the context of requirements engineering, our focus laid on the alignment and diversity of requirements engineering practices, supported by RIMs (Paper E). A certain level of alignment is considered necessary to facilitate coordination across boundaries and adhere to standards, but also diversity is needed to support the heterogeneous needs of disciplines and their methods. Alignment is supported by RIMs by allowing stakeholders to define entity types and relationships, attributes, consistency checks, maturity levels, and Definition of Done criteria. Diversity is enabled by generic relationships, the creation of subtypes, free text fields, and diverse ways of managing backlogs and projects. Diversity is required in early phases, whereas alignment becomes more important when a product is released (see finding 4).

Main finding 7. Alignment and diversity

The balance between alignment and diversity is challenging when managing knowledge for inter-team coordination. Aligned methods facilitate coordination across boundaries, integration, and adherence to standards, whereas diversity is needed to support the heterogeneous needs of disciplines and their methods. RIMs can help to support both alignment and diversity.

1.6.2 RQ2: Guidelines to Manage Knowledge

Several of our papers present guidelines or principles to manage externalized knowledge for inter-team coordination. In the following, we synthesize our findings to answer **RQ2**: *What guidelines can help practitioners to manage knowledge for inter-team coordination in large-scale agile development?*

The guidelines presented in our papers can be grouped into six main areas. Figure 1.6 depicts an overview of these areas (blue boxes) with actors (people icons) and supporting tooling (a network of computers, indicating that a traceable, potentially distributed source of information should be used). The first area is concerned with an analysis of stakeholders, including their needs, goals, interests, and activities. The second area relates to the scoping of externalized knowledge, i.e., what knowledge should be externalized, what levels of abstraction and granularity should be chosen for artifacts and trace links, and what the purpose and audience of artifacts and trace links should be. A key aspect of this area is to understand what artifacts (can) serve as boundary objects. Based on these activities, supporting coordination mechanisms around living boundary objects (3.) can be devised. Living boundary objects should be defined upfront with a lightweight approach, refined using emerging aspects, and coordination can be supported by groups of representatives. While our main topic is concerned with managing knowledge for inter-team coordination, also guidance for intra-team activities was found to be important, to ensure that decisions taken on an inter-team level are reflected in the work of individual teams. Coordination mechanisms around locally relevant artifacts (4.) are strongly connected to face-to-face communication and traceability to boundary objects. It helps to use a traceable source of information to manage knowledge, e.g., using a distributed tool infrastructure. Indicating the benefit of knowledge management and enforcing an appropriate level of rigor (5.) is another area in which we specified guidelines. The last area is concerned with alignment and diversity (6.), for which we recommend focusing on alignment on a high level and diversity on a low level. We describe the six areas in the following.

1. Stakeholder analysis: An analysis of stakeholders and their needs was found to be beneficial in several of our papers. Knowledge management for inter-team coordination can only be successfully conducted with a good understanding of the involved participants. When creating an architecture description, understanding the stakeholders is crucial to define the scope (Paper C).

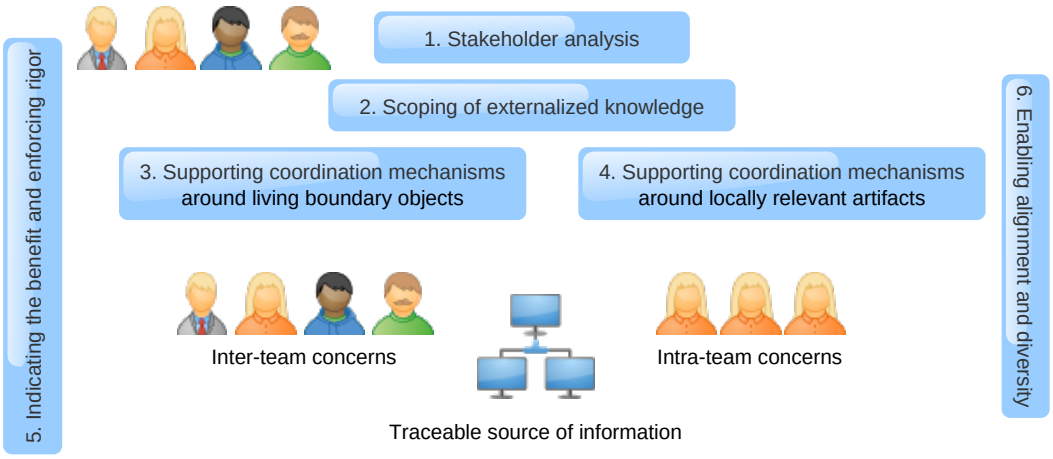

Figure 1.6: Areas of guidelines to manage externalized knowledge for inter-team coordination (RQ2)

Key stakeholders also need to be known to create the required level of alignment (Paper E).

With respect to traceability, we advise putting stakeholders' information needs and goals at the center (Paper B). Typical concerns related to traceability can be covered with a lightweight, developer-driven approach or a formal, requirements-centered approach. Based on an understanding of stakeholders' tasks and needs, practitioners can balance the effort and benefit of traceability management per role.

2. Scoping of externalized knowledge: It is important to select the right scope and level of detail when externalizing knowledge and creating artifacts or trace links. Living boundary objects can be key enablers of inter-team coordination, but need to be kept up to date over time. As many organizations make use of legacy artifacts and trace links, one guideline is to identify existing boundary object candidates and locally relevant artifacts and evaluate their relevance and usage at frequent intervals (Paper A). Stakeholders from different areas should be involved in this activity to cover the perspectives of the whole organization and ensure that artifacts and trace links can provide value.

When creating an architecture description, the purpose and audience need to be clarified (Paper C). This information can be used to choose the scope of an architecture description, in particular, its level of abstraction and what elements should be included. We advise distinguishing between the present and future perspectives of a system in order to be transparent about what system at what point in time an architecture description holds for. While our guideline is to minimize the number of elements in the upfront architecture description, we recommend including elements that are relevant across team boundaries. These guidelines ensure that the description can be leveraged as a living boundary object, while other concerns might be left up to individual teams. Interfaces are typical elements specified in architecture descriptions and we suggest selecting carefully what interfaces need to be managed centrally (Paper D). Early stage interfaces, for instance, should be left up to the teams.

Interfaces should be defined on a high level of abstraction, if possible, so that they can be used in various contexts (Paper D).

3. Supporting coordination mechanisms around living boundary objects: Living boundary objects can help to establish a common understanding of a system across team borders, but need to be kept up to date and traced to other artifacts. We suggest establishing a group of representatives for each boundary object (Papers A and C), in which changes can be discussed and implemented. For architectural knowledge, we recommend involving stakeholders with architecture expertise to establish interfaces that set sufficiently stable boundaries so that teams can work autonomously (Paper D).

Boundary objects should be defined upfront with a lightweight and flexible approach, so that they can be stable and recognizable. Our advice is to assess artifacts in the early stages with a few users before they become more widely used and change becomes more difficult (Papers D and E). Ideally, architects are then integrated into agile teams to capture emerging aspects (Paper C). Using these guidelines, living boundary objects can be established that are traced to other artifacts, kept up to date, and are in line with the changing needs of an organization.

4. Supporting coordination mechanisms around locally relevant artifacts: Locally relevant artifacts play a role for a team, but are not central for inter-team coordination. However, the connection to living boundary objects should be kept. We advise using one source of information and making it traceable (Paper C). Traceability can enable change propagation and notification across boundaries and ensure that changes in boundary objects are reflected in locally relevant artifacts and in the final system (Paper B).

Generally, locally relevant artifacts can be produced as late as possible and only when they are actually needed. Stakeholders need to see the artifacts' relevance and use. We suggest working towards generating locally relevant artifacts and making them reusable (Paper A). Following these principles, practitioners can ensure that locally relevant artifacts are perceived as relevant, kept up to date, and are consistent with boundary objects.

5. Indicating the benefit and enforcing rigor wherever needed: We recommend indicating the benefit of knowledge management and convincing practitioners to invest in knowledge management mainly because they are intrinsically motivated. Indicating the benefit of knowledge management is easiest if artifacts and trace links can be used to support software and systems engineering activities. We advise investing in ways to leverage externalized knowledge and stressing its use (Paper A). Some artifacts and trace links should be centrally managed with a sufficient level of rigor. Central vehicle interfaces, in particular, should be changed with controlled mechanisms and undergo versioning (Paper D). Moreover, we suggest striving for a rigorous culture with respect to traceability maintenance (Paper B) to improve trace link quality. However, generally, training and communication should be favored over strong restrictions (Paper E), so that rigor is only enforced wherever it is indispensable.

6. Enabling alignment and diversity: Large-scale organizations benefit from aligned practices to facilitate integration, but also diverse practices that support the individual needs of agile teams and disciplines. We recommend creating a generally applicable RIM that only requires practitioners to fill in the amount of information for requirements that is absolutely necessary (Paper E). When special procedures for safety, testing, or releases are needed for some requirements, separate requirement types or attributes can be beneficial. A common high-level structure is recommended to support alignment while different modeling styles are beneficial on a lower level (Paper E).

These guidelines relate to the previously mentioned guidelines concerning rigor. Traceability should be rigorously maintained to support the establishment of living and traceable boundary objects for inter-team coordination. It helps to establish aligned practices for aspects that require high data quality and benefit from a high level of rigor. Following these guidelines, sufficiently aligned practices enable the management of living boundary objects, but also the needs of different disciplines are acknowledged and diverse methods are supported.

1.6.3 RQ3: Tool Solutions to Manage Knowledge

We conceived several concepts related to tool support for knowledge management in this PhD thesis. In this section, we answer **RQ3**: *How can tool solutions support practitioners to manage knowledge for inter-team coordination in large-scale agile development?* The findings are mainly based on Papers A and F. The developed concepts for tool solutions are (1.) support for the *visualization and identification of boundary objects*, (2.) *flexible versioning*, (3.) *data quality and change management support*, as well as (4.) *organizational traceability*.

1. Visualization and identification of boundary objects: To support inter-team coordination, one of our guidelines relates to identifying boundary object candidates, i.e., existing artifacts that potentially help to create a common understanding across team borders. In Paper A, we suggest identifying boundary object candidates by leveraging trace links and analyzing what artifacts are referred to by other artifacts from different areas. The data should be visualized in appropriate ways and validated with experts. Adding information about the creators and users of information is essential to understand which artifacts are referred to by multiple teams.

Today's tools do not have a first-class representation of boundary objects. Figure 1.7 shows what the visualization of boundary objects and locally relevant artifacts could look like. Boundary objects are indicated in gray boxes on the left and locally relevant artifacts in blue boxes. The visualization in the example figure is created from the perspective of Anna, a developer of the *Adaptive Cruise Control (ACC)* component, for which requirements, design models, and a testing area are stored. Two boundary objects are shown: The vertical boundary object *Functional Requirement FReq-23*, with Rachel as the boundary object owner, and the horizontal boundary object *Velocity signal*, with Ben as the owner. Trace links are depicted on a high level: a requirement is traced to the functional requirement and an element in the design models is

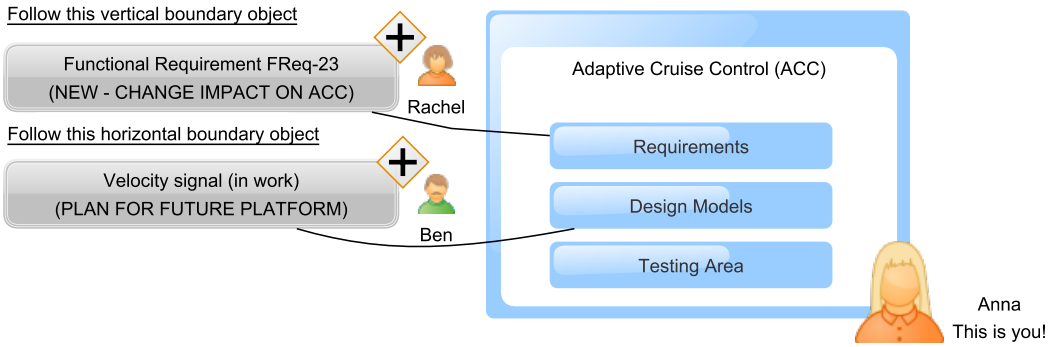

Figure 1.7: Visualization of boundary object candidates and locally relevant artifacts in a tool

traced to the velocity signal. Anna has the possibility to follow the boundary objects by clicking on the plus symbol and will be notified about changes.

2. Flexible versioning: In Paper A, we describe that boundary objects should be stable and recognizable and therefore undergo versioning and releases. Locally relevant artifacts can be unversioned or have the status "*in work*" for a longer period of time. In Paper F, we describe that strict (fine-grained) versioning of all changes would be beneficial for some artifacts, whereas others could be loosely versioned (i.e., only when users manually decide to create a new version of something). Mild versioning of a locally relevant artifact could create intermediate versions whenever a change to that artifact is related to a major change to a boundary object, but rely on loose versioning otherwise. Today's tools typically either rely on loose versioning or strict versioning for all artifacts. Having more fine-granular solutions can help to enforce rigor wherever necessary and support lightweight knowledge management when working with locally relevant artifacts.

Figure 1.8 depicts examples of boundary objects and locally relevant artifacts with their version numbers. A red *SV* indicates that an artifact undergoes strict versioning, *MV* indicates mild versioning, and an *LV* symbol indicates loose versioning. The boundary object *FReq-23* is a functional requirement versioned with a strict versioning approach. Design models are loosely versioned, as they are locally relevant artifacts in an early phase. *Req-4543* is a locally relevant artifact in version 2 and traced to *FReq-23*. It is mildly versioned, which means that intermediate versions are created whenever a change is created in connection with a change to the boundary object *FReq-23*.

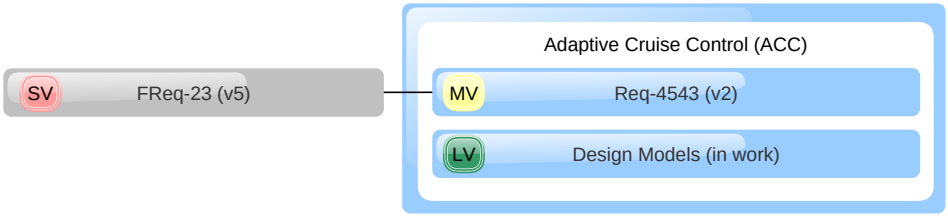

Figure 1.8: Visualization of proposed versioning mechanisms

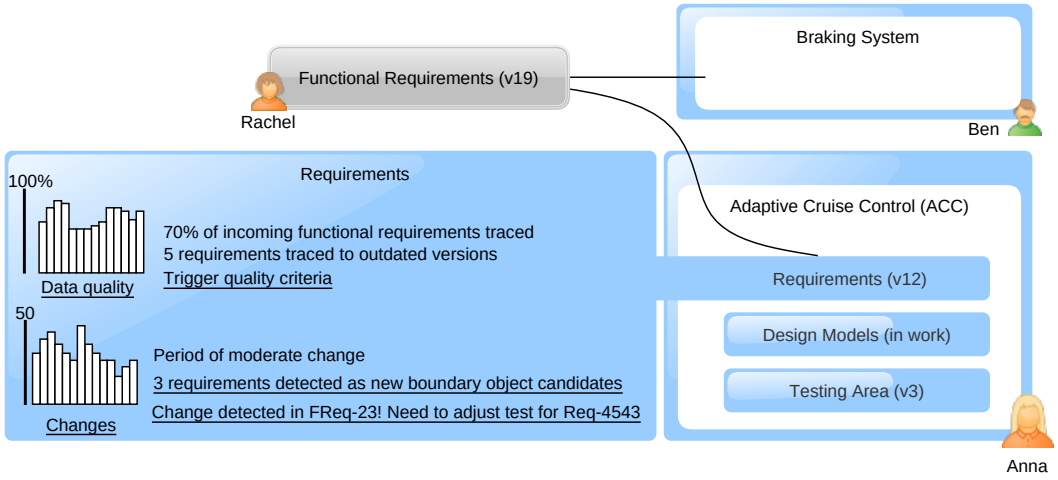

Figure 1.9: Visualization of mechanisms to support data quality and change management

3. Data quality and change management support: Living boundary objects needs to be up to date and of high quality, so that their benefit can be realized. According to our findings, users should be supported in defining data quality needs and evolving them over time throughout the lifecycle (Paper F). Moreover, functions to support change notification and capturing the rationales of changes can be beneficial. In Figure 1.7, short comment texts are shown for changed artifacts (e.g., "*PLAN FOR FUTURE PLATFORM*"), as well as plus signs that users can click on to receive notifications about future changes to boundary objects.

Figure 1.9 visualizes examples of mechanisms to support data quality and change management. In the figure, detailed information on data quality and changes is shown for the *Requirements* of the *ACC* component. For data quality and changes, histograms are shown on the left, indicating the percentage of fulfilled data quality criteria and the number of changes in a certain time interval (with 50 being the maximum number displayed). With respect to completeness, 70% of the incoming functional requirements are currently traced to requirements of the component. As for version consistency, 5 requirements are traced to outdated versions of the functional requirements. Users can click on links to receive more information on data quality and adjust data quality criteria.

Information about changes is shown below the data quality information. Currently, there is a period of moderate change. Three requirements are detected as new boundary object candidates. The user can click on this information to confirm them as boundary objects and get further change notifications. Moreover, a change has been detected in the functional requirement *FReq-23* and the need to adjust the test for an ACC requirement is indicated. Ideally, users should be provided with mechanisms to configure what information is shown in such a visualization and potentially propagate changes automatically.

4. Organizational traceability: As indicated in previous examples of visualizations (e.g., Figure 1.7 and 1.9), the owners and potentially affected stakeholders of artifacts should be indicated. Based on that information, networks of stakeholders and groups can be created and organizational traceability established. Organizational traceability should be supported by allowing users to capture organizational structures, groups, and roles, as well as responsible stakeholders for boundary objects. As responsibilities change over time (e.g., because a locally relevant artifact might become a boundary object), it should be evolved in a semi-automatic way. For instance, if a locally relevant artifact becomes a boundary object, the tool might suggest that the creator of that locally relevant artifact shall become the boundary object owner.

1.7 Discussion

In this thesis, we aim to improve how practitioners can manage knowledge for inter-team coordination in large-scale agile development. This section briefly summarizes our findings, discusses them with related work, and describes implications for research and practice. Table 1.2 presents a summary of the points that will be mentioned in this discussion, including references to related work. These points refer both to the current state of knowledge management for inter-team coordination and to solutions.

What is the current state of managing knowledge for inter-team coordination?

We found that knowledge in large-scale agile contexts is commonly externalized in artifacts and trace links with heterogeneous characteristics. The heterogeneity of artifacts and the variety of tools are recognized challenges with traceability [66]. The role of artifacts in large-scale agile development has received increasing attention in the last years and several researchers have published inventories of artifacts [41, 43]. The artifacts we focus on are not pure software development artifacts and the data for Paper A was collected while the case companies were transitioning to agile methods. Product backlogs, status boards, or other typical agile artifacts did not play a central role in our study. Recently, the role of the product backlog as a boundary object has been investigated [89], with the conclusion that it is an "*informal model of work to be done*" and not a requirements specification. Our participants likely focused on artifacts that served for long-term knowledge management in the study in Paper A. Only one survey respondent mentioned the word "*backlog*" briefly in a comment.

The importance of quality of externalized knowledge in agile development has been found crucial also in related studies, especially with respect to consistency (e.g., [93, 94]). Moreover, we could confirm several ways in which externalized knowledge is leveraged in practice, e.g., for change management [47,60] or to support coordination within a company [41]. We found that quality needs to be on a high level to leverage the benefit of knowledge management.

Generally, individuals can be intrinsically motivated to invest in the creation and maintenance of artifacts and trace links because they see the benefit of these activities. An alternative is to establish a rigorous culture, which

Table 1.2: Summary of points in discussion

	Summary of points and references to related work
Heterogeneity	Artifacts and trace links have heterogeneous characteristics [41, 43, 66]. Our participants focused on long-term knowledge management and not on backlogs [89]. Stakeholder analysis is crucial to determine what artifacts and links should be created [48, 90]. Boundary object candidates can be identified by analyzing trace links, which complements previously proposed analysis approaches [91, 92].
Quality	Quality of artifacts and trace links is crucial [93, 94]. Visualization features can improve the quality of trace links [95, 96]. Quality metrics should be tailored [97] to particular artifacts and trace links.
Use of knowledge	Externalized knowledge can be used for several activities [41,47,60], but high quality of artifacts and trace links is needed.
Lifecycle	Changes to artifacts and trace links are inevitable [98,99]. Artifacts and organizations typically become more stable and formal over time [100]. We recommend defining boundary objects upfront [101] with a lightweight approach. Tailored change management practices are essential and can be supported by traceability [47, 102, 103].
Supporting coordination mechanisms	We recommend creating groups of representatives for boundary objects [40, 59]. Locally relevant artifacts should be flexibly managed, traced, and made reusable. The effort of tracing should be minimized [104]. Organizational traceability can support knowledge management and coordination mechanisms [14, 58, 95].
Benefit and rigor	The lack of perceived benefit when creating artifacts and trace links is a recognized problem [105,106]. Living boundary objects can constitute new benefits. We advise creating sufficiently rigorous practices, in line with recommendations for Just-Enough-Discipline [45].
Alignment and diversity	Aligned/standardized practices are considered desirable for certain purposes and situations [26, 107, 108], but also diversity in practices is needed [109,110]. We advise establishing alignment on a high level and supporting diversity for low-level concerns. Information models need to be tailored to specific contexts [87, 111]. Our guidelines for RIMs can help to support both diversity and alignment.

includes defining the practices that should be used, as well as communicating and following them consistently. In line with our findings, Klein stated that a rigorous culture requires Just-Enough-Discipline (JED) to be in place, a concept describing the balance between the spontaneous, fluid exchange of knowledge and structured, codified, and institutional approaches to knowledge management [45].

Previous studies have also confirmed our observed issues with the lack of benefit of traceability management [105]. Berry et al. [106] described four ways of solving the problem of the lack of benefit of a document to its producer: external or internal benefits should be produced—by increasing reward, decreasing pain, increasing the internal benefit, or decreasing the lack of benefit. Our findings indicate that the concept of living boundary objects can decrease the pain of individuals by focusing on the most central artifacts for inter-team

coordination and constitute new internal benefits, for instance, because investing in traceability can enable collaboration. Focused studies could investigate these aspects in further detail, e.g., by implementing the suggested tool solutions and assessing their effects on collaboration, perceived motivation, and data quality.

Our findings with respect to the lifecycle of externalized knowledge in large-scale agile development relate to the identified inevitability of changes in artifacts, e.g., requirements or interfaces [98, 99]. We investigated the need for change to information models and devised models for the lifecycle of RIMs and the OADI cycle of traceability. The importance of tailoring such information models to the needs of specific contexts has been identified [87,111], but we are unaware of other studies focusing on how to customize them and ensure that changing stakeholders' needs are met throughout the lifecycle. In Paper C, one of the findings is that stakeholders aim to keep interfaces stable when the position in the lifecycle increases and they are increasingly used by components and teams. This finding is in line with Mintzberg's findings that the formalization of an organization's behavior grows with its age and size [100]. With growing stability, also change management mechanisms become more formalized and rigid. We confirmed these points and concluded that with an increasing distance between teams, also more rigorous and formal coordination mechanisms are established. Living boundary objects can help to enable coordination on central aspects for which a shared understanding is necessary and ensure that these objects are kept up to date.

Our findings regarding alignment and diversity are motivated by the interdisciplinary nature of many large-scale development contexts. While standardization of processes is desirable in some situations, it has been found crucial to tailor processes to the contexts where they are applied [107]. Process diversity is an important part of software development, especially in large organizations [109]. In practice, hybrid development methods, combining practices of several development methods, are widely used [110]. Boehm and Turner suggest integrating agile practices into traditional processes by minimizing the parts for which alignment is enforced and carefully identifying situations in which more rigor is needed [108]. Our contributions in Paper E show why and how a RIM can support the balance between alignment and diversity. With appropriate practices, a RIM can serve as a living boundary object and facilitate inter-team coordination in large-scale agile contexts.

What guidelines and tool solutions can help practitioners?

In order to address practical challenges, we identified the importance of establishing living boundary objects. While the concept of boundary objects has existed before, we identified important properties that they should have to serve as living artifacts and support inter-team coordination. Living boundary objects create a common understanding across team borders, are kept up to date, and traced to and from other artifacts, so that inter-team coordination in large-scale agile development can be supported.

Our guidelines regarding stakeholder analysis and scoping of externalized knowledge are in line with related work. For example, architecture knowledge management requires understanding stakeholders' concerns and analyz-

ing what to capture in architecture descriptions [90]. Moreover, the planning and management of a traceability strategy require determining stakeholders' needs [48]. These activities can help to identify what artifacts and trace links are important and establish living boundary objects.

In practice, large-scale companies typically have legacy systems, documentation, and processes. One of our suggestions for tool solutions involves leveraging trace links to identify candidates for boundary objects. Trace links make dependencies between artifacts explicit, which is especially important when change impact analysis needs to be performed to support software and systems evolution [91]. It has been suggested to analyze artifacts with their numbers of trace links before [92]: an unusually high number of incoming or outgoing trace links might indicate that an artifact has a too broad scope or that the traceability information model should be adjusted. Our findings indicate that analyzing trace links cannot only help to tailor the traceability information model, but also to identify potential boundary objects and support inter-team coordination.

Living boundary objects are at the center of inter-team coordination. We suggest defining elements concerning multiple teams (e.g., interfaces or information models) upfront, but refining them over time with emerging aspects. This guideline can help to address challenges in the area of agile architecting [101]. Defining living boundary objects early on helps create to "*islands of stability*" that are required for agile teams to work autonomously and flexibly [26]. Communities of practice or other groups of representatives are suggested as additional coordination mechanisms around living boundary objects [40, 59]. These groups support the exchange of tacit knowledge as a complement to the management of explicit knowledge in living boundary objects.

Relying on strong knowledge networks and boundary spanners between teams are other mechanisms for knowledge coordination [14, 58]. Studies on knowledge networks relate to the proposed tooling for organizational traceability. Organizational traceability can be enabled by sociograms, which have been studied in the context of traceability before [95]. Teams and organizational structures can change over time, which is why practitioners should keep track of networks of artifacts, users, and responsibilities. Changes to artifacts should be reflected in tool solutions (e.g., when a boundary object becomes locally relevant or vice versa), to update change management features or tune data quality levels. With respect to change management, several tool solutions have been proposed, many of which are related to traceability [102,103]. Combining traceability and change management can support knowledge integration and facilitate development activities (e.g., impact analysis and program comprehension) [47].

Locally relevant artifacts should be created late, so that documents are created only when they are actually needed, and traced to boundary objects. Trace links within a team can be managed with a flexible approach and face-to-face communication. Previous work has recommended minimizing tracing and only focusing on crucial trace links that help to improve quality and meet stakeholders' expectations [104]. Artifacts should be made reusable and generated, as with prescriptive models [112]. Stakeholders are more motivated to keep prescriptive artifacts up to date, as they need to be consistent and of

high quality for the final software or system to work (Paper A).

A key issue is to motivate individuals to invest in knowledge management by indicating the benefit or fostering a rigorous culture. We suggest prioritizing training and communication rather than strong restrictions (Paper E) and establishing rigorous change management mechanisms around artifacts impacting many stakeholders (e.g., central vehicle interfaces, Paper D). We propose to visualize the level of data quality (e.g., regarding consistency and completeness). There exist several approaches to visualize traceability-related data that help to maintain trace links and increase consistency [95, 96]. Our findings suggest that configurable levels of data quality can be beneficial, depending on the current phase in the development lifecycle and depending on whether an artifact is a boundary object or locally relevant. Data quality is crucial to ensure that artifacts and trace links can be leveraged for practical purposes [97]. Version consistency is a special kind of consistency that we found to be relevant in our participating companies. To allow for more flexible management, we suggest three levels of versioning: strict, mild, and loose versioning. Generally, living boundary objects should be stable and recognizable (especially in later lifecycle phases), whereas locally relevant artifacts can be loosely versioned and managed. Information can also be made visible in tools, so that users can detect changes, propagate them, and keep boundary objects as living artifacts [47].

We advise establishing alignment on a high level, with a common structure and a path of minimal information. Diversity can be beneficial in early phases and for low-level information. Standard practices regarding architecture and management can help to implement agile methods in systems engineering [108]. Following these guidelines, the aforementioned need for stability around agile teams [26] can be fulfilled. Aligning an organization on living boundary objects and enabling diverse ways of managing locally relevant artifacts can support agility at scale, as well as flexible knowledge management for inter-team coordination.

The Concept of Living Boundary Objects for Large-Scale Agile Development

Several of our proposed guidelines and solutions relate to the cornerstones of agile development. In the following, we elaborate on the concepts that we studied to arrive at this thesis' main proposition: to leverage living boundary objects to support agile inter-team coordination at scale. Based on our main findings and related work, we discuss relationships between concepts of knowledge management in large-scale agile development and present them to motivate how living boundary objects can serve as key enablers of agility. We refer to Bertrand Meyer's agile principles [113], which have been presented in a critical and well-established book on agility. In particular, we identified that our research on knowledge management for inter-team coordination in large-scale agile development relates to three principles: let the team self-organize, produce frequent working iterations, and accept change [113]. Figure 1.10 shows these agile principles, selected concepts investigated in this thesis, and their relationships. The selected agile principles are shown in gray boxes (labeled as "*agile*"). Supporting concepts are shown in orange boxes and main

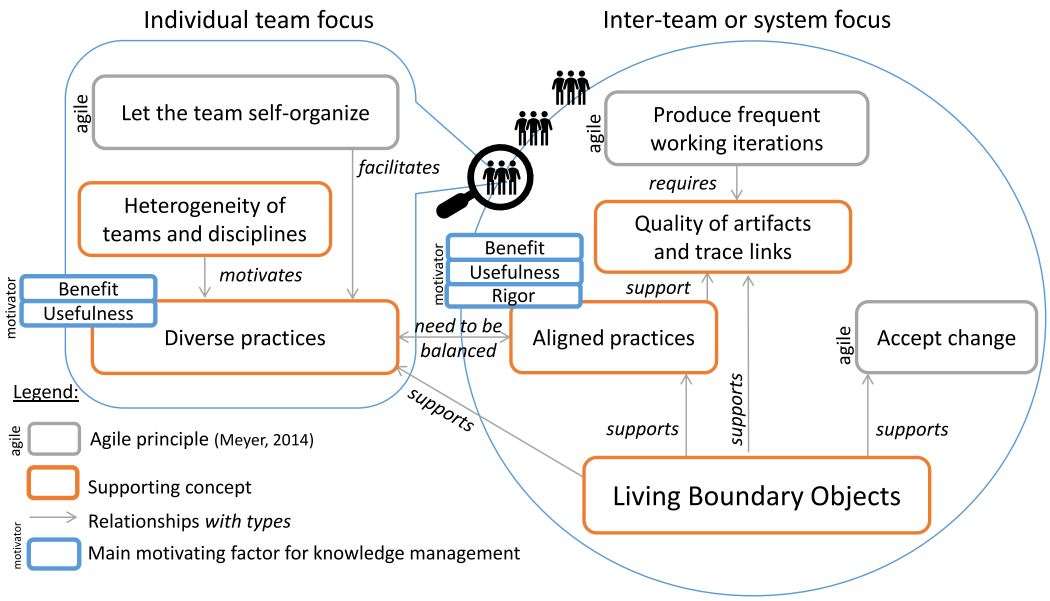

Figure 1.10: Overview of selected concepts of knowledge management for agile inter-team coordination and agile principles [113]

motivating factors for knowledge management are depicted in blue boxes. The left part of the figure depicts the focus on individual teams, whereas the right part shows the inter-team or system focus in large-scale agile organizations.

An agile principle related to the individual team focus is to let the team self-organize. Self-organization or empowerment facilitates the emergence of diverse practices. In the principles of the agile manifesto, this point is phrased as follows: "*At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*" [21]. In this thesis, we study the phenomenon of diverse practices in large-scale systems engineering companies, in which heterogeneous teams and disciplines collaborate. The heterogeneity further motivates the need for diverse practices.

In the part related to the inter-team or system focus, an agile principle is to produce frequent working iterations and support frequent integration of software [113]. To implement this principle, the quality of artifacts and trace links is essential, e.g., regarding consistency or completeness [114]. To facilitate integration, the components developed by different teams need to be consistent. Moreover, a complete safety analysis is often required by safety standards before software can be deployed to systems. In large-scale agile development, artifacts and trace links are typically created, maintained, and used by multiple teams. Our findings indicate that aligned practices of teams can contribute towards creating high-quality artifacts and trace links that, in turn, can be used to support integration and the production of frequent working iterations. However, aligned practices and diverse practices need to be balanced. Another agile principle is to accept change and deal with evolution in large-scale agile development. According to Meyer, accepting change includes supporting and planning for it by producing extendable systems [113].

Living boundary objects are facilitators of several of the described concepts and address the relevant agile principles. By focusing on the lifecycle perspective of artifacts and trace links, we analyzed how living boundary objects can

be used to accept and support change. Living boundary objects are kept up to date and traced to and from other artifacts, which facilitates change impact analysis and other activities related to change management [114]. We also described how RIMs and TIMs can be changed over time to address practitioners' evolving needs. Our guidelines in Paper C are concerned with designing a lightweight upfront architecture and supporting change by capturing emerging elements. Similarly, living boundary objects can also be leveraged to support quality needs. Living boundary objects serve as a common reference and are traced to and from other artifacts, which supports practitioners in re-establishing consistency and validity after changes happen. Moreover, the suggested tool solutions for living boundary objects in Paper F can support data quality needs and evolve them with stakeholders' needs. Furthermore, living boundary objects can support alignment by establishing a common identity across sites. We suggest supporting diverse practices, as living boundary objects allow for interpretive flexibility and heterogeneous ways of managing knowledge on a lower level. We found that the main motivating factors for individual teams with diverse practices are the benefit and usefulness of knowledge management: if the benefits outweigh the effort, teams see a point in investing in knowledge management. A positive cost-benefit balance is crucial to motivate teams. While the general goal on the inter-team level is also to aim for high benefit and usefulness, it can be beneficial to establish a sufficiently rigorous culture for aligned practices.

In summary, our findings indicate how living boundary objects can support agile inter-team coordination in large-scale organizations. In particular, we have discussed how they can support diverse and aligned practices, achieve traceability and high quality of artifacts and trace links, and facilitate change. The understanding of concepts and their relations can help to motivate future research and improve industrial practices, as we present in the following.

Implications for research and practice

This thesis addresses the topic of knowledge management for inter-team coordination, which is one of the challenges on the research agenda on large-scale agile development [5]. To complement existing studies on tacit knowledge management, e.g., related to Communities of Practice [40], we focused especially on explicit knowledge, externalized in artifacts and trace links. The recent publications of related studies on explicit knowledge management in agile development, especially focusing on artifacts, indicate the timeliness of this topic [42, 43, 89]. This thesis contributes to the body of knowledge of inter-team coordination in large-scale agile development in several ways. We present empirical evidence of how knowledge is externalized in heterogeneous artifacts and trace links, as well as how they are changed and used over time in large-scale agile development. Practitioners aim to minimize the documentation effort in agile development by focusing on only absolutely crucial artifacts and trace links. We found that some artifacts can act as boundary objects and create a shared understanding between agile teams while being adaptable to the needs of individual teams and disciplines. We coined the concept of *living boundary objects*, which are traced to other artifacts, kept up to date, and serve for inter-team coordination. Using living boundary objects, the effort

of knowledge management can be kept to a minimum and agile organizations can focus on providing customer value. Our guidelines and concepts for tool solutions can help to establish sufficiently aligned practices in different teams but also empower agile teams to use diverse methods that fulfill the needs of team members and disciplines.

This thesis can inspire researchers that aim to conceive tailored methods and tool solutions for knowledge management in large-scale agile development. Until now, a majority of the methods in the field of software engineering do not consider the importance of differentiating between artifacts that are used for inter-team coordination and those relevant for one team. Our findings indicate that more tailored approaches are needed that take changes throughout the development lifecycle into account and also consider involved organizational groups. We suggest solutions that distinguish between locally relevant artifacts and living boundary objects and provide adjustable data quality levels, change management support, collaborative traceability management, and flexible versioning mechanisms.

When we conducted the first studies of this thesis, a majority of the collaborating companies were still in the early phases of the adoption of agile methods. Pinpointing the need for managing externalized knowledge in large-scale agile development and introducing the concept of living boundary objects helps practitioners to reflect on what documentation is needed and prioritize artifacts and links that are used for inter-team coordination. The suggested tool solutions and guidelines can help tool suppliers and large-scale companies to establish practices that are fit for purpose and create new benefits for individuals investing in knowledge management.

1.8 Conclusion

Agile methods have become widely spread in the last decades, not only in small and co-located teams but also in large-scale software and systems engineering. A common challenge in large-scale agile development is the coordination between agile teams. In this thesis, we focus on how knowledge is and should be managed to support inter-team coordination in large-scale agile development.

The focus of agile development lies on providing customer value and increasing development speed, which is why agile organizations aim to reduce the effort of documentation. Starting with exploratory studies on artifacts and trace links, we identified the crucial role of boundary objects, which are artifacts that create a shared understanding between teams, while being adaptable to the needs of teams and disciplines. We found the necessity to investigate further aspects of knowledge management in depth: architectural knowledge, traceability management, and requirements and traceability information models. We focused on how artifacts and trace links are changed throughout their lifecycles and presented specific guidelines. These guidelines help to focus the knowledge management effort on absolutely crucial artifacts and trace links, namely those needed for inter-team coordination. Moreover, we scrutinized the need to create sufficient alignment in large-scale agile development, while enabling agile teams to conceive tailored and diverse practices that fulfill their individual needs. The identified practical needs can be addressed with bound-

ary objects that fulfill a number of key properties. To this end, we coined the concept of *living boundary objects*, i.e., boundary objects that are traced to other artifacts, kept up to date, and serve for inter-team coordination.

We presented several solutions to support knowledge management for inter-team coordination. We advise performing a stakeholder analysis to determine the desirable scope of artifacts and trace links. The level of abstraction, purpose, and audience of artifacts and trace links should correspond to stakeholders' needs. We recommend managing boundary objects in groups of representatives of several teams, whereas coordination mechanisms for locally relevant artifacts can be conceived by individual teams. Tooling should support organizational traceability to identify what artifacts are relevant for stakeholders from different teams and keep track of responsibilities. We suggest using existing trace links to identify boundary object candidates in a tool. Living boundary objects should be created upfront with a lightweight approach, undergo strict versioning, and be refined over time with emerging aspects. Artifacts that are developed and used within one team can be flexibly created and loosely versioned. Change management support and adjustable dashboards indicating data quality levels can help to ensure that stakeholders' quality needs are fulfilled.

The empirical evidence presented in this PhD thesis contributes to the body of knowledge of large-scale agile development. Our findings shed light on the state of the practice of knowledge management for agile inter-team coordination at scale, which can be utilized both by researchers and practitioners. The presented guidelines can be used as an instrument to establish successful knowledge management practices in large-scale organizations, considering the distinct role of living boundary objects for inter-team coordination. The provided concepts for tool support further facilitate these endeavors and can be used to determine, refine, and utilize data quality levels, change management, and versioning mechanisms. We envision a future in which living boundary objects are used to actively exchange knowledge between teams and disciplines, stress the benefits of traceability management, and help practitioners to focus on the important aspects to align large-scale agile organizations on.

Bibliography

[1] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[2] D. X. Houston, "Agility beyond software development," in *Proceedings of the International Conference on Software and System Process (ICSSP'14)*, 2014, pp. 65–69.

[3] CollabNet VersionOne, "The 13th annual state of agile report," 2019, accessed on August 10, 2019. [Online]. Available: https://www.stateofagile.com/

[4] P. Abrahamsson, K. Conboy, and X. Wang, "'lots done, more to do': the current state of agile systems development research," *European Journal of Information Systems*, vol. 18, no. 4, pp. 281–284, Aug. 2009.

[5] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim, "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation," *Empirical Software Engineering*, vol. 23, no. 1, pp. 490–520, Feb. 2018.

[6] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016.

[7] K. H. Rolland, B. Fitzgerald, T. Dingsøyr, and K.-J. Stol, "Problematizing agile in the large: Alternative assumptions for large-scale agile development," in *Proceedings of the 37th International Conference on Information Systems (ICIS'16)*, 2016, pp. 1–21.

[8] M. Paasivaara, C. Lassenius, and V. T. Heikkilä, "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?" in *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'12)*, 2012, pp. 235–238.

[9] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *Proceedings of the Future of Software Engineering (FoSE'07)*, May 2007, pp. 188–198.

[10] A. Scheerer, T. Hildenbrand, and T. Kude, "Coordination in large-scale agile software development: A multiteam systems perspective," in *47th Hawaii International Conference on System Sciences (HICSS'14)*, Jan. 2014, pp. 4780–4788.

[11] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb, "Team knowledge and coordination in geographically distributed software development," *Journal of Management Information Systems*, vol. 24, no. 1, pp. 135–169, 2007.

[12] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38, no. 3, pp. 69–81, Mar. 1995.

[13] D. Turk, F. Robert, and B. Rumpe, "Assumptions underlying agile software-development processes," *Journal of Database Management*, vol. 16, no. 4, pp. 62–87, 2005.

[14] N. B. Moe, D. Šmite, A. Šāblis, A.-L. Börjesson, and P. Andréasson, "Networking in a large-scale distributed agile project," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'14)*, 2014.

[15] B. Ramesh, "Process knowledge management with traceability," *IEEE Software*, vol. 19, no. 3, pp. 50–52, 2002.

[16] M. Zahedi, M. Shahin, and M. A. Babar, "A systematic review of knowledge sharing challenges and practices in global software development," *International Journal of Information Management*, vol. 36, no. 6, Part A, pp. 995–1019, 2016.

[17] M. Kajko-Mattsson, "Problems in agile trenches," in *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*, 2008.

[18] A. Rüping, *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects*, 1st ed. Wiley Publishing, 2003.

[19] S. L. Star and J. R. Griesemer, "Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39," *Social Studies of Science*, vol. 19, no. 3, pp. 387–420, 1989.

[20] P. Abrahamson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," *VTT Publications*, 2002.

[21] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001, accessed on March 3, 2020. [Online]. Available: http://www.agilemanifesto.org/

[22] T. S. Schmidt, S. Weiss, and K. Paetzold, "Expected vs. real effects of agile development of physical products: Apportioning the hype," in *Proceedings of the International Design Conference (DESIGN'18)*, vol. 5, 2018, pp. 2121–2132.

[23] D. J. Reifer, F. Maurer, and H. Erdogmus, "Scaling agile methods," *IEEE Software*, vol. 20, no. 4, pp. 12–14, 2003.

[24] T. Dingsøyr and N. B. Moe, "Towards principles of large-scale agile development," in *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen, Eds. Cham: Springer International Publishing, 2014, pp. 1–8.

[25] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What is large in large-scale? a taxonomy of scale for agile software development," in *Proceedings of the International Conference on Product-Focused Software Process Improvement (PROFES'14)*, 2014, pp. 273–276.

[26] R. L. Nord, I. Ozkaya, and P. Kruchten, "Agile in distress: Architecture to the rescue," in *Proceedings of the 15th International Conference on Agile Software Development (XP'14)*. Springer International Publishing, 2014, pp. 43–57.

[27] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional, 2007.

[28] C. Larman and B. Vodde, *Large-Scale Scrum: More with LeSS*, ser. Addison-Wesley Signature Series (Cohn). Pearson Education, 2016.

[29] C. Ebert and M. Paasivaara, "Scaling agile," *IEEE Software*, vol. 34, no. 6, pp. 98–103, 2017.

[30] F. Calefato and C. Ebert, "Agile collaboration for distributed teams," *IEEE Software*, vol. 36, no. 1, pp. 72–78, 2019.

[31] L. G. Terveen, "Overview of human-computer collaboration," *Knowledge-Based Systems*, vol. 8, no. 2-3, pp. 67–81, 1995.

[32] T. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys (CSUR)*, vol. 26, no. 1, pp. 87–119, 1994.

[33] G. A. Okhuysen and B. A. Bechky, "10 coordination in organizations: an integrative perspective," *Academy of Management annals*, vol. 3, no. 1, pp. 463–502, 2009.

[34] V. Stray, N. B. Moe, and R. Hoda, "Autonomous agile teams: Challenges and future directions for research," in *Proceedings of the 19th International Conference on Agile Software Development: Companion (XP'18)*. ACM, 2018.

[35] A. H. Van De Ven, A. L. Delbecq, and R. Koenig, "Determinants of coordination modes within organizations," *American Sociological Review*, vol. 41, no. 2, pp. 322–338, 1976.

[36] N. J. Cooke, E. Salas, J. A. Cannon-Bowers, and R. J. Stout, "Measuring team knowledge," *Human Factors*, vol. 42, no. 1, pp. 151–173, 2000.

[37] D. P. Wallace, *Knowledge management: Historical and cross-disciplinary themes*. Libraries unlimited, 2007.

[38] B. D. Newman and K. W. Conrad, "A framework for characterizing knowledge management methods, practices, and technologies," in *Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (PAKM'00)*, 2000, pp. 1–11.

[39] K. Dalkir, "Knowledge management in theory and practice," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 10, pp. 2083–2083, 2011.

[40] T. Kähkönen, "Agile methods for large organizations – building communities of practice," in *Proceedings of the Agile Development Conference (ADC'04)*. IEEE, 2004, pp. 2–10.

[41] G. Wagenaar, S. Overbeek, G. Lucassen, S. Brinkkemper, and K. Schneider, "Working software over comprehensive documentation – rationales of agile teams for artefacts usage," *Journal of Software Engineering Research and Development*, vol. 6, no. 1, p. 7, Jul. 2018.

[42] D. Méndez Fernández, W. Böhm, A. Vogelsang, J. Mund, M. Broy, M. Kuhrmann, and T. Weyer, "Artefacts in software engineering: a fundamental positioning," *Software & Systems Modeling*, vol. 18, no. 5, pp. 2777–2786, Oct. 2019.

[43] J. M. Bass, "Artefacts and agile method tailoring in large-scale offshore software development programmes," *Information and Software Technology*, vol. 75, pp. 1–16, 2016.

[44] A. B. Graham and V. G. Pizzo, "A question of balance: Case studies in strategic knowledge management," *European Management Journal*, vol. 14, no. 4, pp. 338–346, 1996.

[45] D. Klein, *The Strategic Management of Intellectual Capital*, ser. Knowledge Reader Series. Butterworth-Heinemann, 1998.

[46] S. Ghobadi, "What drives knowledge sharing in software development teams: A literature review and classification framework," *Information & Management*, vol. 52, no. 1, pp. 82–97, 2015.

[47] K. Mohan, P. Xu, L. Cao, and B. Ramesh, "Improving change management in software development: Integrating traceability and software configuration management," *Decision Support Systems*, vol. 45, no. 4, pp. 922–936, 2008.

[48] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. Maletic, and P. Mäder, "Traceability fundamentals," in *[49]*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 3–22.

[49] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability*. Springer-Verlag London Limited, 2012.

[50] M. Callon, "Some elements of a sociology of translation: Domestication of the scallops and the fishermen of st brieuc bay," in *Power, Action and Belief: A New Sociology of Knowledge*, J. Law, Ed. London: Routledge & Kegan Paul, 1986, ch. 5.

[51] B. Latour, *Science in action: How to follow scientists and engineers through society*. Harvard University Press, 1987.

[52] A. Mol, "Actor-network theory: Sensitive terms and enduring tensions," *Kölner Zeitschrift für Soziologie und Sozialpsychologie. Sonderheft*, vol. 50, pp. 253–269, 2010.

[53] E. Bjarnason and H. Sharp, "The role of distances in requirements communication: a case study," *Requirements Engineering*, vol. 22, no. 1, pp. 1–26, Mar. 2017.

[54] R. Abraham, *Guidelines for Architecture Models as Boundary Objects*. Cham: Springer International Publishing, 2017, pp. 193–210.

[55] P. Xu, "Coordination in large agile projects," *Review of Business Information Systems (RBIS)*, vol. 13, no. 4, 2009.

[56] G. C. Bowker and S. L. Star, *Sorting things out: classification and its consequences*. Cambridge, Mass.: MIT Press, 1999.

[57] N. Levina and E. Vaast, "The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems," *MIS Quarterly*, vol. 29, no. 2, pp. 335–363, 2005.

[58] D. Šmite, N. B. Moe, A. Šāblis, and C. Wohlin, "Software teams and their knowledge networks in large-scale software development," *Information and Software Technology*, vol. 86, pp. 71–86, 2017.

[59] D. Šmite, N. B. Moe, G. Levinta, and M. Floryan, "Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations," *IEEE Software*, vol. 36, no. 2, pp. 51–57, Mar. 2019.

[60] J. Cleland-Huang, "Traceability in agile projects," in *[49]*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 265–275.

[61] A. Espinoza and J. Garbajosa, "A study to support agile methods more effectively through traceability," *Innovations in Systems and Software Engineering*, vol. 7, no. 1, pp. 53–69, 2011.

[62] D. Redmiles, A. Van Der Hoek, B. Al-Ani, T. Hildenbrand, S. Quirk, A. Sarma, R. Filho, C. de Souza, and E. Trainer, "Continuous coordination-a new paradigm to support globally distributed software development projects," *Wirtschaftsinformatik*, vol. 49, no. 1, pp. 28–38, 2007.

[63] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. I. Maletic, "The grand challenge of traceability (v1.0)," in *[49]*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 343–409.

[64] H. Nyrud and V. Stray, "Inter-team coordination mechanisms in large-scale agile," in *Proceedings of the XP'2017 Scientific Workshops*, 2017, pp. 1–6.

[65] S. Gayer, A. Herrmann, T. Keuler, M. Riebisch, and P. O. Antonino, "Lightweight traceability for the agile architect," *Computer*, vol. 49, no. 5, pp. 64–71, 2016.

[66] S. Maro, J.-P. Steghöfer, and M. Staron, "Software traceability in the automotive domain: Challenges and solutions," *Journal of Systems and Software*, vol. 141, pp. 85–110, 2018.

[67] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering.* Hoboken, New Jersey: John Wiley & Sons, Inc., 2012.

[68] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," *Guide to Advanced Empirical Software Engineering*, pp. 285–311, 2008.

[69] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[70] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, pp. 75–105, 2004.

[71] R. K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods)*, 4th ed. Sage Publications, 2008.

[72] S. E. Sim, "Evaluating the evidence: Lessons from ethnography," in *Proceedings of the Workshop on Empirical Studies of Software Maintenance*, 1999, pp. 66–70.

[73] R. Wohlrab, P. Pelliccione, E. Knauss, and S. C. Gregory, "The problem of consolidating RE practices at scale: An ethnographic study," in *Proceedings of the 24th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'18)*, 2018, pp. 155–170.

[74] J. W. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 3rd ed. London, England: Sage Publications Ltd., 2008.

[75] E.-M. Ihantola and L.-A. Kihn, "Threats to validity and reliability in mixed methods accounting research," *Qualitative Research in Accounting & Management*, vol. 8, no. 1, pp. 39–58, 2011.

[76] D. S. Cruzes and T. Dybå, "Recommended steps for thematic synthesis in software engineering," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM'11)*, no. 7491, 2011, pp. 275–284.

[77] D. S. Cruzes, T. Dybå, P. Runeson, and M. Höst, "Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1634–1665, Dec. 2015.

[78] E. Guba and Y. Lincoln, *Fourth Generation Evaluation*. SAGE Publications, 1989.

[79] C. Ebert and J. Favaro, "Automotive software," *IEEE Software*, vol. 34, no. 3, pp. 33–39, 2017.

[80] L. Pareto, P. Eriksson, and S. Ehnebom, "Architectural descriptions as boundary objects," in *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS'10)*, 2010, pp. 406–419.

[81] M. Hummel, C. Rosenkranz, and R. Holten, "The role of communication in agile systems development: An analysis of the state of the art," *Business and Information Systems Engineering*, vol. 5, no. 5, pp. 343–355, 2013.

[82] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," *Information and Software Technology*, vol. 54, no. 1, pp. 16–40, 2012.

[83] R. Heldal, P. Pelliccione, U. Eliasson, J. Lantz, J. Derehag, and J. Whittle, "Descriptive vs prescriptive models in industry," in *Proceedings of the 19th International Conference on Model Driven Engineering Languages and Systems (MODELS'16)*, 2016, pp. 216–226.

[84] N. Ali, S. Baker, R. O'Crowley, S. Herold, and J. Buckley, "Architecture consistency: State of the practice, challenges and requirements," *Empirical Software Engineering*, vol. 23, no. 1, pp. 224–258, Feb. 2018.

[85] B. H. Cheng and J. M. Atlee, "Research directions in requirements engineering," in *Proceedings of the Future of Software Engineering (FoSE'07)*. IEEE, May 2007, pp. 285–303.

[86] M. Weber and J. Weisbrod, "Requirements engineering in automotive development—experiences and challenges," in *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)*, Sep. 2002, pp. 331–340.

[87] A. Davis, *Just enough requirements management: where software development meets marketing*, 1st ed. Addison-Wesley Professional, 2013.

[88] C. Neumüller and P. Grünbacher, "Automating software traceability in very small companies: A case study and lessons learned," in *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, Sep. 2006, pp. 145–156.

[89] T. Sedano, P. Ralph, and C. Péraire, "The product backlog," in *Proceedings of the 41th International Conference on Software Engineering (ICSE'19)*, May 2019, pp. 200–211.

[90] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, and M. A. Babar, "10 years of software architecture knowledge management: Practice and future," *Journal of Systems and Software*, vol. 116, pp. 191–205, 2016.

[91] P. Mäder and O. Gotel, "Ready-to-use traceability on evolving projects," in *[49]*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012, pp. 173–194.

[92] P. Mäder, *Rule-based Maintenance of Post-Requirements Traceability*, ser. MV Wissenschaft. MV-Verlag, 2010.

[93] F. G. de Oliveira Neto, J. Horkoff, E. Knauss, R. Kasauli, and G. Liebel, "Challenges of aligning requirements engineering and system testing in large-scale agile: A multiple case study," in *Proceedings of the 25th IEEE International Requirements Engineering Conference Workshops (REW'17)*, Sep. 2017, pp. 315–322.

[94] J. Medeiros, A. Vasconcelos, C. Silva, and M. Goulão, "Quality of software requirements specification in agile projects: A cross-case analysis of six companies," *Journal of Systems and Software*, vol. 142, pp. 171–194, 2018.

[95] C. R. B. de Souza, T. Hildenbrand, and D. Redmiles, "Toward visualization and analysis of traceability relationships in distributed and offshore software development projects," in *Software Engineering Approaches for Offshore and Outsourced Development*, B. Meyer and M. Joseph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 182–199.

[96] A. Marcus, X. Xie, and D. Poshyvanyk, "When and how to visualize traceability links?" in *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'05)*, 2005, pp. 56–61.

[97] P. Rempel and P. Mäder, "A quality model for the systematic assessment of requirements traceability," in *Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE'15)*, Aug. 2015, pp. 176–185.

[98] L. S. Wheatcraft, "9.2.2 Everything you wanted to know about interfaces, but were afraid to ask," *INCOSE International Symposium*, vol. 20, no. 1, pp. 1132–1149, 2010.

[99] A. M. Davis, N. Nurmuliani, S. Park, and D. Zowghi, "Requirements change: What's the alternative?" in *Proceedings of the 32nd Annual*

IEEE International Computer Software and Applications Conference, 2008, pp. 635–638.

[100] H. Mintzberg, *Mintzberg on Management: Inside Our Strange World of Organizations*. Simon and Schuster, 1989.

[101] M. Waterman, J. Noble, and G. Allan, "How much up-front? a grounded theory of agile architecture," in *Proceedings of the 37th International Conference on Software Engineering (ICSE'15)*, 2015, pp. 347–357.

[102] J. Cleland-Huang, C. K. Chang, and M. Christensen, "Event-based traceability for managing evolutionary change," *Transactions on Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[103] S. Jayatilleke and R. Lai, "A systematic review of requirements change management," *Information and Software Technology*, vol. 93, pp. 163–185, 2018.

[104] G. Fournier, "Essential traceability," in *Essential Testing: A Use Case Driven Approach*. CreateSpace Independent Publishing Platform, 2007, ch. 19.

[105] P. Arkley and S. Riddle, "Overcoming the traceability benefit problem," in *Proceedings of the 13th IEEE International Requirements Engineering Conference (RE'05)*, 2005, pp. 385–389.

[106] D. M. Berry, K. Czarnecki, M. Antkiewicz, and M. Abdelrazik, "The problem of the lack of benefit of a document to its producer (PotLoBoaDtiP)," in *Proceedings of the IEEE International Conference on Software Science, Technology and Engineering (SWSTE'16)*, Jun. 2016, pp. 37–42.

[107] M. Lycett, R. D. Macredie, C. Patel, and R. J. Paul, "Migrating agile methods to standardized development practice," *Computer*, vol. 36, no. 6, pp. 79–85, 2003.

[108] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations," *IEEE Software*, vol. 22, no. 5, pp. 30–39, 2005.

[109] M. Lindvall and I. Rus, "Process diversity in software development," *IEEE Software*, vol. 17, no. 4, pp. 14–18, 2000.

[110] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. G. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann, "What are hybrid development methods made of? an evidence-based characterization," in *Proceedings of the IEEE/ACM International Conference on Software and System Processes (ICSSP'19)*, 2019, pp. 105–114.

[111] D. Méndez Fernández and S. Wagner, "Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany," *Information and Software Technology*, vol. 57, pp. 616–643, 2015.

[112] U. Eliasson, R. Heldal, P. Pelliccione, and J. Lantz, "Architecting in the automotive domain: Descriptive vs prescriptive architecture," in *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA'15)*, 2015, pp. 115–118.

[113] B. Meyer, *Agile! The Good, the Hype and the Ugly.* Springer Publishing Company, Incorporated, 2014.

[114] D. A. Meedeniya, I. D. Rubasinghe, and I. Perera, "Software artefacts consistency management towards continuous integration: A roadmap," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, pp. 100–110, 2019.

[115] R. Turner, "Toward agile systems engineering processes," *Cross Talk, The Journal of Defense Software Engineering*, 2007.

[116] R. Haberfellner and O. de Weck, "10.1.3 Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering," *INCOSE International Symposium*, vol. 15, no. 1, pp. 1449–1465, 2005.

[117] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *Proceedings of the International Conference on Systems, Man, and Cybernetics (SMC'17)*, Oct. 2017, pp. 2075–2080.

[118] W. Scacchi, "Managing software engineering projects: A social analysis," *IEEE Transactions on Software Engineering*, no. 1, pp. 49–59, 1984.

[119] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, "Boundary objects in agile practices: Continuous management of systems engineering artifacts in the automotive domain," in *Proceedings of the International Conference on Software and System Process (ICSSP'18)*, 2018, pp. 31–40.

[120] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study," *Empirical Software Engineering*, vol. 15, no. 6, pp. 654–693, 2010.

[121] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Stahl, "The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at Ericsson," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM'13)*, 2013, pp. 348–356.

[122] M. Kuhrmann, D. Méndez Fernández, and M. Gröber, "Towards artifact models as process interfaces in distributed software projects," in *Proceedings of the 8th IEEE International Conference on Global Software Engineering (ICGSE'13)*, Aug. 2013, pp. 11–20.

[123] D. Méndez Fernández, B. Penzenstadler, M. Kuhrmann, and M. Broy, "A meta model for artefact-orientation: Fundamentals and lessons

learned in requirements engineering," in *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS'10)*, 2010, pp. 183–197.

[124] D. Méndez Fernández, W. Böhm, A. Vogelsang, J. Mund, M. Broy, M. Kuhrmann, and T. Weyer, "Artefacts in software engineering: What are they after all?" *arXiv preprint arXiv:1806.00098*, 2018.

[125] S. Voigt, J. von Garrel, J. Müller, and D. Wirth, "A study of documentation in agile software projects," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'16)*, 2016, pp. 4:1–4:6.

[126] J. K. Blomkvist, J. Persson, and J. Åberg, "Communication through boundary objects in distributed agile teams," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*, 2015, pp. 1875–1884.

[127] B. A. Bechky, "Sharing meaning across occupational communities: The transformation of understanding on a production floor," *Organization Science*, vol. 14, no. 3, pp. 312–330, 2003.

[128] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, "Collaborative traceability management: Challenges and opportunities," in *Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16)*, Sep. 2016, pp. 216–225.

[129] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Proceedings of the Future of Software Engineering (FoSE'07)*, 2007, pp. 55–71.

[130] P. Pelliccione, E. Knauss, R. Heldal, M. Ågren, P. Mallozzi, A. Alminger, and D. Borgentun, "Automotive architecture framework: The experience of Volvo Cars," *Journal of Systems Architecture*, vol. 77, 2017.

[131] U. Eliasson, R. Heldal, E. Knauss, and P. Pelliccione, "The need of complementing plan-driven requirements engineering with emerging communication: Experiences from Volvo Car Group," in *Proceedings of the IEEE 23rd International Requirements Engineering Conference (RE'15)*, Aug. 2015, pp. 372–381.

[132] International Organization for Standardization, "Road vehicles – functional safety," *ISO26262:2011*, Nov. 2011.

[133] VDA QMC Working Group 13 / Automotive SIG, "Automotive spice process assessment / reference model," *Automotive SPICE:2017*, vol. 3.1, 2017.

[134] P. Diebold, T. Zehler, and D. Richter, "How do agile practices support automotive SPICE compliance?" in *Proceedings of the International Conference on Software and System Process (ICSSP'17)*, 2017, pp. 80–84.

[135] R. Abraham, "Enterprise architecture artifacts as boundary objects - a framework of properties," in *Proceedings of the 21st European Conference on Information Systems (ECIS'13)*, 2013.

[136] J. Lave, "Situating learning in communities of practice," *Perspectives on socially shared cognition*, vol. 2, pp. 63–82, 1991.

[137] E. Wenger, R. A. McDermott, and W. Snyder, *Cultivating communities of practice: A guide to managing knowledge*. Harvard Business Press, 2002.

[138] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Dec. 2009.

[139] M. C. Tremblay, A. R. Hevner, and D. J. Berndt, "The use of focus groups in design science research," in *Integrated Series in Information Systems*. Springer US, 2010, pp. 121–143.

[140] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 5–55, 1932.

[141] R. Atkinson and J. Flint, "Accessing hidden and hard-to-reach populations: Snowball research strategies," *Social research update*, vol. 33, no. 1, pp. 1–4, 2001.

[142] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, 1st ed. Prentice Hall PTR, 2001.

[143] J. A. Maxwell, *Qualitative Research Design: An Interactive Approach*, ser. Applied Social Research Methods. SAGE Publications, 2012.

[144] C. R. Prause and Z. Durdik, "Architectural design and documentation: Waste in agile development?" in *Proceedings of the International Conference on Software and System Process (ICSSP'12)*, 2012, pp. 130–134.

[145] B. Selic, "Agile documentation, anyone?" *IEEE Software*, vol. 26, no. 6, pp. 11–12, 2009.

[146] P. R. Carlile, "A pragmatic view of knowledge and boundaries: Boundary objects in new product development," *Organization Science*, vol. 13, no. 4, pp. 442–455, 2002.

[147] G. K. Hanssen, A. F. Yamashita, R. Conradi, and L. Moonen, "Maintenance and agile development: Challenges, opportunities and future directions," in *Proceedings of the 25th IEEE International Conference on Software Maintenance (ICSM'09)*, 2009, pp. 487–490.

[148] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, K. Trektere, F. McCaffery, G. Vahid, M. Felderer, O. Linssen, E. Hanser, and C. Prause, "Hybrid software development approaches in practice: A european perspective," *IEEE Software*, vol. 36, no. 4, pp. 20–31, 2018.

[149] R. L. Nord and J. E. Tomayko, "Software architecture-centric methods and agile development," *IEEE Software*, vol. 23, no. 2, pp. 47–53, Mar. 2006.

[150] O. Gotel, J. Cleland-Huang, J. Huffman Hayes, A. Zisman, A. Egyed, P. Grunbacher, and G. Antoniol, "The quest for ubiquity: A roadmap for software and systems traceability research," in *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE'12)*, 2012, pp. 71–80.

[151] G. Spanoudakis and A. Zisman, "Software traceability: A roadmap," in *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing, 2005, vol. 3, pp. 395–428.

[152] P. Mäder and A. Egyed, "Do developers benefit from requirements traceability when evolving and maintaining a software system?" *Empirical Software Engineering*, vol. 20, no. 2, pp. 413–441, 2015.

[153] V. Sinha, B. Sengupta, and S. Chandra, "Enabling collaboration in distributed requirements management," *IEEE Software*, vol. 23, no. 5, pp. 52–61, 2006.

[154] M. Franzago, D. Di Ruscio, I. Malavolta, and H. Muccini, "Collaborative model-driven software engineering: a classification framework and a research map," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1146–1175, 2018.

[155] I. Santiago, Á. Jiménez, J. M. Vara, V. De Castro, V. A. Bollati, and E. Marcos, "Model-driven engineering as a new landscape for traceability management: A systematic literature review," *Information and Software Technology*, vol. 54, pp. 1340–1356, 2012.

[156] F. Furtado and A. Zisman, "Trace++: A traceability approach to support transitioning to agile software engineering," in *Proceedings of the 24th International Requirements Engineering Conference (RE'16)*, 2016, pp. 66–75.

[157] A. de Lucia, R. Oliveto, and G. Tortora, "IR-based traceability recovery processes: An empirical comparison of one-shot and incremental processes," in *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE'08)*, 2008, pp. 39–48.

[158] N. Ali, Z. Sharafl, Y. Gueheneuc, and G. Antoniol, "An empirical study on requirements traceability using eye-tracking," in *Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM'12)*, 2012, pp. 191–200.

[159] S. Sengupta, A. Kanjilal, and S. Bhattacharya, "Requirement traceability in software development process: An empirical approach," in *Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping (RSP'08)*, 2008, pp. 105–111.

[160] L. Klimpke and T. Hildenbrand, "Towards end-to-end traceability: Insights and implications from five case studies," in *Proceedings of the 4th International Conference on Software Engineering Advances (ICSEA'09)*, 2009, pp. 465–470.

[161] P. Mäder, O. Gotel, and I. Philippow, "Motivation matters in the traceability trenches," in *Proceedings of the 17th IEEE International Requirements Engineering Conference (RE'09)*, 2009, pp. 143–148.

[162] E. Bouillon, P. Mäder, and I. Philippow, "A survey on usage scenarios for requirements traceability in practice," in *Proceedings of the 19th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'13)*, 2013, pp. 158–173.

[163] O. Gotel and A. C. W. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the 1st IEEE International Conference on Requirements Engineering (RE'94)*, 1994, pp. 94–101.

[164] B. Ramesh, "Factors influencing requirements traceability practice," *Communications of the ACM*, vol. 41, no. 12, pp. 37–44, 1998.

[165] M. Rath, J. Rendall, J. L. C. Guo, J. Cleland-Huang, and P. Mäder, "Traceability in the wild: Automatically augmenting incomplete trace links," in *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*, 2018, pp. 834–845.

[166] M. Rahimi and J. Cleland-Huang, "Evolving software trace links between requirements and source code," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2198–2231, Aug. 2018.

[167] S. Maro, A. Anjorin, R. Wohlrab, and J.-P. Steghöfer, "Traceability maintenance: Factors and guidelines," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16)*, 2016.

[168] A. Demuth, R. Kretschmer, A. Egyed, and D. Maes, "Introducing traceability and consistency checking for change impact analysis across engineering tools in an automation solution company: An experience report," in *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'16)*, Oct. 2016, pp. 529–538.

[169] M. C. Figueiredo and C. R. De Souza, "Wolf: Supporting impact analysis activities in distributed software development," in *Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'12)*, 2012, pp. 40–46.

[170] J. Helming, M. Koegel *et al.*, "Traceability-based change awareness," in *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09)*, 2009, pp. 372–376.

[171] D. Strašunskas, "Traceability in collaborative systems development from lifecycle perspective," in *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'02)*, 2002, pp. 54–60.

[172] J. Cleland-Huang, O. C. Z. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: Trends and future directions," in *Proceedings of the Future of Software Engineering (FoSE'14)*, 2014, pp. 55–69.

[173] L. A. Palinkas, S. M. Horwitz, C. A. Green, J. P. Wisdom, N. Duan, and K. Hoagwood, "Purposeful sampling for qualitative data collection and analysis in mixed method implementation research," *Administration and Policy in Mental Health and Mental Health Services Research*, vol. 42, no. 5, pp. 533–544, 2015.

[174] M. D. Myers and M. Newman, "The qualitative interview in IS research: Examining the craft," *Information and Organization*, vol. 17, no. 1, pp. 2–26, 2007.

[175] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[176] J. Maxwell, "Understanding and validity in qualitative research," *Harvard Educational Review*, vol. 62, no. 3, pp. 279–301, 1992.

[177] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards, "Requirements traceability: Theory and practice," *Annals of Software Engineering*, vol. 3, no. 1, pp. 397–415, Jan. 1997.

[178] A. Kannenberg and H. Saiedian, "Why software requirements traceability remains a challenge," *The Journal of Defense Software Engineering*, vol. 22, no. 7, pp. 14–19, 2009.

[179] V. Kirova, N. Kirby, D. Kothari, and G. Childress, "Effective requirements traceability: Models, tools, and practices," *Bell Labs Technical Journal*, vol. 12, no. 4, pp. 143–157, 2008.

[180] H. U. Asuncion, F. François, and R. N. Taylor, "An end-to-end industrial software traceability tool," in *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'07)*, 2007, pp. 115–124.

[181] M. C. Panis, "Successful deployment of requirements traceability in a commercial engineering organization...really," in *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10)*, Sep. 2010, pp. 303–307.

[182] K. Schneider, *Experience and Knowledge Management in Software Engineering*. Springer, 2009.

[183] A. Averbakh, "Light-weight experience collection in distributed software engineering," Ph.D. dissertation, Leibniz Universität Hannover, 2014.

[184] S. K. Sundaram, J. Hayes Huffman, A. Dekhtyar, and E. A. Holbrook, "Assessing traceability of software engineering artifacts," in *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10)*, 2010, pp. 313–335.

[185] B. Sengupta, S. Chandra, and V. Sinha, "A research agenda for distributed software development," in *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, 2006, pp. 731–740.

[186] N. Sekitoleko, F. Evbota, E. Knauss, A. Sandberg, M. Chaudron, and H. H. Olsson, "Technical dependency challenges in large-scale agile software development," in *Proceedings of the International Conference on Agile Software Development (XP'14)*, 2014, pp. 46–61.

[187] S. F. Königs, G. Beier, A. Figge, and R. Stark, "Traceability in systems engineering — review of industrial practices, state-of-the-art technologies and new research solutions," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 924–940, 2012.

[188] K. Jaber, B. Sharif, and C. Liu, "A study on the effect of traceability links in software maintenance," *IEEE Access*, vol. 1, pp. 726–741, 2013.

[189] P. Rempel, P. Mäder, and T. Kuschke, "An empirical study on project-specific traceability strategies," in *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE'13)*, 2013, pp. 195–204.

[190] K. Schneider, "Rationale as a by-product," in *Rationale Management in Software Engineering*, A. H. Dutoit, R. McCall, I. Mistrík, and B. Paech, Eds. Springer Berlin Heidelberg, 2006, pp. 91–109.

[191] M. Lang and J. Duggan, "A tool to support collaborative software requirements management," *Requirements Engineering*, vol. 6, no. 3, pp. 161–172, Oct. 2001.

[192] M. Shaw and P. Clements, "The golden age of software architecture," *IEEE Software*, vol. 23, no. 2, pp. 31–39, Mar. 2006.

[193] M. A. Babar, "Supporting the software architecture process with knowledge management," in *Software Architecture Knowledge Management: Theory and Practice*, M. Ali Babar, T. Dingsøyr, P. Lago, and H. van Vliet, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 69–86.

[194] P. Abrahamsson, M. A. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Software*, vol. 27, no. 2, pp. 16–22, Mar. 2010.

[195] G. Booch, "An architectural oxymoron," *IEEE Software*, vol. 27, no. 5, pp. 96–96, Sep. 2010.

[196] T. Mens and S. Demeyer, Eds., *Software Evolution*. Springer-Verlag Berlin Heidelberg, 2008, vol. 1.

[197] M. Feilkas, D. Ratiu, and E. Jürgens, "The loss of architectural knowledge during system evolution: An industrial case study," in *Proceedings of the 24rd IEEE/ACM International Conference on Automated Software Engineering (ICPC'09)*, May 2009, pp. 188–197.

[198] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," *Handbook of Software Engineering and Knowledge Engineering*, vol. 1, pp. 329–380, 2001.

[199] B. A. Kitchenham and S. L. Pfleeger, *Personal Opinion Surveys*. London: Springer London, 2008, pp. 63–92.

[200] F. J. Fowler, Jr., *Survey Research Methods*, 3rd ed., ser. Applied Social Research Methods. Thousand Oaks, CA: SAGE Publications, 2002.

[201] ISO/IEC, "ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description," 2011.

[202] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, 1st ed. Springer-Verlag Berlin Heidelberg, 2012.

[203] P. Kruchten, "What do software architects really do?" *Journal of Systems and Software*, vol. 81, no. 12, pp. 2413–2416, 2008.

[204] M. Waterman, "Agility, risk, and uncertainty, part 1: Designing an agile architecture," *IEEE Software*, vol. 35, no. 2, pp. 99–101, Mar. 2018.

[205] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. A. Babar, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, no. 3, pp. 352–370, 2010.

[206] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.

[207] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, Oct. 1992.

[208] E. Poort, "Just enough anticipation: Architect your time dimension," *IEEE Software*, vol. 33, no. 6, pp. 11–15, Nov. 2016.

[209] M. Riaz, M. Sulayman, and H. Naqvi, "Architectural decay during continuous software evolution and impact of 'design for change' on software architecture," in *Proceedings of the International Conference on Advanced Software Engineering and Its Applications (ASEA'09)*, 2009, pp. 119–126.

[210] L. De Silva and D. Balasubramaniam, "Controlling software architecture erosion: A survey," *Journal of Systems and Software*, vol. 85, no. 1, pp. 132–151, 2012.

[211] B. Weitzel, D. Rost, and M. Scheffe, "Sustaining agility through architecture: Experiences from a joint research and development laboratory," in *Proceedings of the 11th Working IEEE/IFIP Conference on Software Architecture (WICSA'14)*, 2014, pp. 53–56.

[212] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, Jun. 2013.

[213] X. Wang, X. Zhou, and L. Jiang, "A method of business and IT alignment based on enterprise architecture," in *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI'08)*, 2008, pp. 740–745.

[214] M. Lankhorst, "6 ways to organize your architecture models (part 1)," *BiZZdesign Blog*, 2018, accessed on February 5, 2019.

[215] D. A. Tamburri and E. D. Nitto, "When software architecture leads to social debt," in *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA'15)*, 2015, pp. 61–64.

[216] R. Weinreich and I. Groher, "The architect's role in practice: From decision maker to knowledge manager?" *IEEE Software*, vol. 33, no. 6, pp. 63–69, Nov. 2016.

[217] R. Britto, D. Smite, and L. Damm, "Software architects in large-scale distributed projects: An Ericsson case study," *IEEE Software*, vol. 33, no. 6, pp. 48–55, Nov. 2016.

[218] S. Frey, L. Charissis, and J. Nahm, "How software architects drive connected vehicles," *IEEE Software*, vol. 33, no. 6, pp. 41–47, Nov. 2016.

[219] R. Wohlrab, E. Knauss, J.-P. Steghöfer, S. Maro, A. Anjorin, and P. Pelliccione, "Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture," *Requirements Engineering*, Nov. 2018.

[220] C. Yang, P. Liang, and P. Avgeriou, "A systematic mapping study on the combination of software architecture and agile development," *Journal of Systems and Software*, vol. 111, pp. 157–184, 2016.

[221] K. Read and F. Maurer, "Issues in scaling agile using an architecture-centric approach: A tool-based solution," in *Proceedings of the 3rd XP/Agile Universe Conference*, 2003, pp. 142–150.

[222] K. Rahmani and V. Thomson, "Managing subsystem interfaces of complex products," *International Journal of Product Lifecycle Management*, vol. 5, no. 1, 2011.

[223] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*, ser. SEI series in software engineering. Pearson Education, 2003.

[224] B. Hookway, *Interface*, ser. Cultural studies. MIT Press, 2014.

[225] D. Hoffman, "On criteria for module interfaces," *IEEE Transactions on Software Engineering*, vol. 16, no. 5, pp. 537–542, May 1990.

[226] J. Lindman, J. Horkoff, I. Hammouda, and E. Knauss, "Emerging perspectives of application programming interface strategy: A framework to respond to business concerns," *IEEE Software*, vol. 37, no. 2, pp. 52–59, Mar. 2020.

[227] F. Bachmann, L. Bass, P. Clements, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, "Documenting software architecture: Documenting interfaces," Software Engineering Institute, Carnegie Mellon University, Tech. Rep., 2002.

[228] B. H. Cheng and J. M. Atlee, "Current and future research directions in requirements engineering," in *Design Requirements Engineering: A Ten-Year Perspective*. Springer, 2009, pp. 11–43.

[229] G. John, M. Hoffmann, M. Weber, M. Nagel, and C. Thomas, "Using a common information model as a methodological basis for a tool-supported requirements management process," *INCOSE International Symposium*, vol. 9, no. 1, pp. 1437–1441, 1999.

[230] A. M. Davis and D. Zowghi, "Good requirements practices are neither necessary nor sufficient," *Requirements Engineering*, vol. 11, pp. 1–3, 2005.

[231] M. Broy, I. H. Krüger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356–373, Feb. 2007.

[232] G. Liebel, M. Tichy, E. Knauss, O. Ljungkrantz, and G. Stieglbauer, "Organisation and communication problems in automotive requirements engineering," *Requirements Engineering*, vol. 23, no. 1, pp. 145–167, 2018.

[233] ISO/IEC TR 11179-2:2019, "Information technology – Metadata registries (MDR) – Part 2: Classification," International Organization for Standardization, Tech. Rep., 2019.

[234] E. Hochmüller, "Requirements classification as a first step to grasp quality requirements," in *Proceedings of the 3rd Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'97)*, 1997, pp. 133–144.

[235] T. Gorschek and C. Wohlin, "Requirements abstraction model," *Requirements Engineering*, vol. 11, no. 1, pp. 79–101, 2006.

[236] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints: A framework for integrating multiple perspectives in system development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31–57, 1992.

[237] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," *Annals of Software Engineering*, vol. 3, no. 1, pp. 101–130, Jan. 1997.

[238] D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, ser. Agile Software Development Series. Addison-Wesley Professional, 2011.

[239] D. Méndez Fernández, K. Lochmann, B. Penzenstadler, and S. Wagner, "A case study on the application of an artefact-based requirements engineering approach," in *Proceedings of the 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE'11)*, 2011, pp. 104–113.

[240] L. C. Rodríguez, M. Mora, M. V. Martin, R. O'Connor, and F. Alvarez, "Process models of sdlcs: comparison and evolution," in *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. IGI Global, 2009, pp. 76–89.

[241] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, "Boundary objects and their use in agile systems engineering," *Journal of Software: Evolution and Process*, vol. 31, no. 5, 2019.

[242] S. L. Star, "Chapter 2 - the structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving," in *Distributed Artificial Intelligence*, L. Gasser and M. N. Huhns, Eds. San Francisco (CA): Morgan Kaufmann, 1989, pp. 37–54.

[243] P. A. Laplante, *Requirements engineering for software and systems*. Auerbach Publications, 2017.

[244] J. M. Bhat, M. Gupta, and S. N. Murthy, "Overcoming requirements engineering challenges: Lessons from offshore outsourcing," *IEEE Software*, vol. 23, no. 5, pp. 38–44, Sep. 2006.

[245] E. Serna M., O. Bachiller S., and A. Serna A., "Knowledge meaning and management in requirements engineering," *International Journal of Information Management*, vol. 37, no. 3, pp. 155–161, 2017.

[246] J. Doerr, B. Paech, and M. Koehler, "Requirements engineering process improvement based on an information model," in *Proceedings of the IEEE International Conference on Requirements Engineering (RE'04)*, 2004, pp. 70–79.

[247] M. Hertzum, "Small-scale classification schemes: A field study of requirements engineering," *Computer Supported Cooperative Work*, vol. 13, no. 1, pp. 35–61, 2004.

[248] N. Niu, S. Brinkkemper, X. Franch, J. Partanen, and J. Savolainen, "Requirements engineering and continuous deployment," *IEEE Software*, vol. 35, no. 2, pp. 86–90, 2018.

[249] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile requirements engineering: A systematic literature review," *Computer Standards & Interfaces*, vol. 49, pp. 79–91, 2017.

[250] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior*, vol. 51, pp. 915–929, Oct. 2015.

[251] M. Kassab, "An empirical study on the requirements engineering practices for agile software development," in *Proceedings of the 40th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA'14)*, 2014, pp. 254–261.

[252] D. Fucci, C. Palomares, X. Franch, D. Costal, M. Raatikainen, M. Stettinger, Z. Kurtanovic, T. Kojo, L. Koenig, A. Falkner, G. Schenner, F. Brasca, T. Männistö, A. Felfernig, and W. Maalej, "Needs and challenges for a platform to support large-scale requirements engineering: A multiple-case study," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'18)*, 2018, pp. 19:1–19:10.

[253] J. L. Boulanger and V. Q. Dao, "Requirements engineering in a model-based methodology for embedded automotive software," in *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, 2008, pp. 263–268.

[254] P. Braun, M. Broy, F. Houdek, M. Kirchmayr, M. Müller, B. Penzenstadler, K. Pohl, and T. Weyer, "Guiding requirements engineering for software-intensive embedded systems in the automotive industry," *Computer Science — Research and Development*, vol. 29, no. 1, pp. 21–43, Feb. 2014.

[255] S. Pearson and A. Saeed, "Information structures for traceability for dependable avionic systems," *Technical Report Series—University of Newcastle upon Tyne, Computing Science*, 1997.

[256] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, and M. Stade, "The crowd in requirements engineering: The landscape and challenges," *IEEE Software*, vol. 34, no. 2, pp. 44–52, Mar. 2017.

[257] QSR International Pty Ltd, "NVivo 12 Pro," 2019. [Online]. Available: https://www.qsrinternational.com/nvivo

[258] R. Tesch, *Qualitative Research: Analysis Types and Software Tools*. London: Falmer Press, 1990.

[259] The R Foundation, "The R project for statistical computing," 2019. [Online]. Available: https://www.r-project.org/

[260] C. Braun and R. Winter, "A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform," in *Enterprise Modelling and Information Systems Architectures*, J. Desel and U. Frank, Eds. Gesellschaft für Informatik, 2005, pp. 64–79.

[261] N. B. Moe, A. Aurum, and T. Dybå, "Challenges of shared decision-making: A multiple case study of agile software development," *Information and Software Technology*, vol. 54, no. 8, pp. 853–865, 2012.

[262] A. Shahrokni, P. Gergely, J. Söderberg, and P. Pelliccione, "Organic evolution of development organizations—an experience report," in *Proceedings of the SAE World Congress and Exhibition - Model-Based Controls and Software Development*, Apr. 2016.

[263] J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-driven engineering practices in industry," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE'11)*, 2011, pp. 633–642.

[264] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh, "A manifesto for model merging," in *Proceedings of the International Workshop on Global Integrated Model Management (GaMMa'06)*, 2006, pp. 5–12.

[265] J. V. Bond, III, G. H. Engelman, J. Ekmark, J. L. Jansson, M. N. Tarabishy, and L. Tellis, "Collision mitigation by braking system," Aug. 2003, US Patent 6,607,255 B2.

[266] F. Kofman, "Lecture slides," *MIT Sloan School of Management, Cambridge, MA*, 1992.

[267] S. Mirarab, A. Hassouna, and L. Tahvildari, "Using bayesian belief networks to predict change propagation in software systems," in *Proceedings of the 15th IEEE International Conference on Program Comprehension (ICPC'07)*, 2007, pp. 177–188.